
# Analysis and Implementation of Time-delay Systems and Networked Control Systems

Bo Wang

Ph. D. Thesis

June 2008

R11

## *Certificate of Research*

*This is to certify that, except where specific reference is made, the work described in this thesis is the result of the candidate's research. Neither this thesis, nor any part of it, has been presented, or is currently submitted, in candidature for any degree at any other University.*

*Signed* ..... *Candidate*

*Date* 30. Jun. 2008

*Signed* ..... *Director of Studies*

*Date* 30/06/2008

# Analysis and Implementation of Time-delay Systems and Networked Control Systems

**by Bo Wang**

Prifysgol Morgannwg
The University of Glamorgan

A thesis submitted in partial fulfilment of
the requirement for the degree of Doctor of Philosophy

June 2008

# Abstract

Systems with delays frequently appear in engineering. The presence of delays makes system analysis and control design much more complicated. Networked control systems where the delays are often random are typical cases of such systems.

For one particular category of time-delays systems, integral processes with dead time (IPDTs), the control limits that a PI controller can achieve are discussed in this thesis. These limits include the region of the control parameters to guarantee the system stability, the control parameters to achieve the given gain and/or phase margins (GPMs), the constraint on achievable gain and phase margins, the performance of setpoint tracking and disturbance rejection. Three types of PI controllers, namely typical PI controller, single tuning-parameter PI controller and PI controller under two-degree-of-freedom (2-DOF) structure, are studied.

In control schemes of the modified Smith predictor (MSP) where the controller usually includes a distributed delay, the system implementation is not trivial because of the inherent hidden unstable poles. This thesis provides an estimation of the minimal number of implementation steps for the distributed delay in linear control laws. This is obtained by solving an inequality with respect to the number of implementation steps. A coarse estimation is given as the initial value to solve the inequality using bisection algorithms. A minimization process as well as some other techniques are also introduced to further improve the estimation.

In networked control systems, the network-transmission delay and data dropout are combinedly represented by a network-induced delay. By designing a data preprocessing mechanism, the network-induced delay can be assigned. Such delay assignment is applied to networked predictive control schemes, which alleviates system stability limits on the network-induced delay. Two stability criteria are given for the

closed-loop system with random network-induced delay, and a resulting implementation algorithm is also provided.

The control and implementation of a magnetic levitation system over the network is studied in this thesis. Firstly, a test-rig which is suitable to implement control over a network is set up. Feedback linearization and direct local linearization methods for the nonlinear MagLev system are presented. In order to improve the control performance, a networked predictive method is employed, where the system model is identified in real-time. Local control and networked control are implemented on this test-rig, including networked predictive control. Model predictive control demonstrates a clear performance advantage over the networked control strategies which does not incorporate compensation for the network-induced delay.

In order to quickly implement networked control systems (NCSs) by simulation or practical application, a MATLAB/Simulink based NCS toolbox is developed. This toolbox incorporates basic parts of a general NCS, that is, network simulation, network interface, plant interface and typical control schemes. With the NCS toolbox, users can focus on the study of new control schemes.

# Acknowledgments

First of all, I would like to express my sincere gratitude to my director of studies, Professor Guoping Liu. He guided me throughout the majority of the work reported in this thesis, and his expertise and support proved invaluable. At several stages, it was his inspiration and example that gave me the confidence and strength to carry on with the work. It was really a great pleasure working under his supervision. What I learned from him will be invaluable for my future career.

I am very grateful to my second supervisor Dr David Rees for his great help and support throughout this research program. His advice and guidance to report the research work, with a measure of clarity, proved invaluable.

I would also like to thank my former director of studies, Dr Qingchang Zhong. He guided me in the earlier part of this work and his help in introducing me to this research area is gratefully acknowledged.

I greatly appreciate the enjoyable discussions during this work with Senchun Chai, Wenshan Hu, Yunbo Zhao, Ximing Sun and Rui Wang.

There are no words that suffice to thank my wife Jiahui Qi for her love and understanding. I am also very grateful to my family in China for their constant support.

# Contents

# List of Figures

# List of Tables

# Notation

| | |
|---|---|
| $\mathbb{R}$ | field of real numbers |
| $j\mathbb{R}$ | imaginary axis |
| Re and Im | real and imaginary parts of complex variables |
| $\in$ | belong to |
| $\subset$ | subset |
| $\square$ | end of proof |
| $A^T$ | transpose of matrix $A$ |
| $A^*$ | conjugate transpose of $A$ |
| $A^{-1}$ | inverse of matrix $A$ |
| $\det A$ | determinant of matrix $A$ |
| $\lambda(A)$ | eigenvalue of matrix $A$ |
| $\sigma(A)$ | singular value of matrix $A$ |
| $\|\cdot\|$ | 2-norm |
| $\|\cdot\|_\infty$ | $H_\infty$-norm |
| $P(s) = \left[\begin{array}{c\|c} A & B \\ \hline C & D \end{array}\right]$ | shorthand for $P(s) = D + C(sI - A)^{-1}B$ |
| $\mathbb{R}(z^{-1}, n)$ | set of polynomials w.r.t $z^{-1}$ with order $n$ and with coefficients in $\mathbb{R}$ |

# Acronyms

| | |
|---|---|
| 2-DOF | two-degree-of-freedom |
| AD | analog to digital |
| CPG | control prediction generator |
| DA | digital to analog |
| FIR | finite impulse response |
| FOPDT | first order plus dead time |
| FSA | finite spectrum assignment |
| GPC | general predictive control |
| GPM | gain and/or phase margin |
| IPDT | integral process with dead time |
| ISE | integral square error |
| ISTE | integral square time multiplied error |
| ITSE | integral time square error |
| LQG | linear quadratic Gaussian |
| LMI | linear matrix inequality |
| LTI | linear time-invariant |
| MagLev | magnetic levitation |
| MSP | modified Smith predictor |
| NDC | network delay compensator |
| NCS | networked control system |
| NPC | networked predictive control |
| NS2 | network simulation 2 |
| PID | proportional-integral-derivative |
| PIO | programmable I/O |

| | |
|---|---|
| PWM | pulse width modulation |
| RLS | recursive least square |
| RTW | real-time workshop |
| SCADA | supervisory control and data acquisition |
| SISO | single-input single output |
| TDS | time-delay system |
| w.r.t. | with respect to |

# Chapter 1

# Introduction

In this chapter, the background of the control, stability analysis and implementation of time-delay systems (TDSs) and networked control systems (NCSs) is reviewed. The main content of each chapter of this thesis is outlined.

## 1.1 Background

Time-delay systems (TDSs) are usually described by retarded functional differential equations or neutral functional differential equations in mathematics (Bellman and Cooke, 1963; Kolmanovskii and Nosov, 1986; Hale and Verduyn-Lunel, 1993). They frequently appear in engineering. One reason is that the physical world is full of delays because the propagation and transmission of information or material involves time lags. Another reason is that time-delay systems are often used to model high-order systems. The delay might be in the input, in the output, in the state, or in the controller of a system. Typical examples of TDSs are chemical processes, communication networks, teleoperation systems and so on. The presence of delays, especially large delays, varying delays or even randoms delays, makes the stability analysis, the control design and the system implementation much more complicated. In recent years, research on this subject has been very active.

### 1.1.1 General time-delay systems

The delay in TDS is generally constant within a finite time interval or often can be treated as being constant. Unlike a rational function, the delay element is infinitely dimensional in nature and, in many cases, it is the source of instability. There are substantial publications that have addressed the stability analysis of TDS. In the time domain, most approaches are based on the Lyapunov-Krasovskii theorem (Krasovskii, 1963; Kolmanovskii and Nosov, 1986) and the Razumikhin theorem (Jankovic, 2001), which are extensions of the Lyapunov stability theory to time-delay systems. As far as linear time-invariant systems, it is convenient to analyze their stability using frequency domain methods. The resulting closed-loop characteristic equation is a quasi-polynomial of $s$ (complex variable) and $e^{-\tau s}$ (denoting time delay $\tau$) and it can be decomposed into real and imaginary parts to check the stability (Pontryagin, 1955). If $e^{-\tau s}$ is substituted with some rational functions, such as the bilinear form, the quasi-polynomial reduces to a polynomial so that the stability analysis becomes easier (Olgac and Sipahi, 2002). A direct method to eliminate the term $e^{-\tau s}$ in a quasi-polynomial was proposed by (Walton and Marshall, 1987).

For some systems which are stable when the delay is set to zero, the frequency sweeping method (Chen, 1995) is used to investigate the maximum delay that is permissible for system stability. An explicit formulation of the maximum delay is obtained in (Chiasson and Abdallah, 2001). For uncertain time-delay systems, $H^\infty$ and $\mu$-type of results (Zhou et al., 1996) can be used to analyze the stability. For a more detailed introduction of recent results about the stability of TDS, refer to Kharitonov (1999); Kolmanovskii et al. (1999); Niculescu (2001); Gu and Niculescu (2003); Richard (2003)

The first major advance in the control of time-delay systems is the celebrated Smith predictor (SP) (Smith, 1957). By introducing a minor feedback loop consisting of a predictor, the controller design problem for a time-delay system is then converted to the one for delay-free system. The classical Smith predictor, however, is not applicable to unstable plants. This motivated the modified Smith predictor (MSP) (Watanabe and Ito, 1981) and the finite-spectrum assignment (Manitius and Olbrot, 1979), which are actually equivalent for some systems (Mirkin and Raskin, 2003). MSP may run into numerical problems for fast stable eigenvalues Meinsma and Zwart (2000). A new predictor, called unified Smith predictor (Zhong and Weiss, 2004), is proposed to overcome this problem.

An integral process with dead time (IPDT) is a typical case of unstable processes, which stands for a large class of industrial processes. Åström et al. (1994) introduced a structure to decouple the disturbance response from the setpoint response. A disturbance-observer-based control scheme is another effective way to control such systems (Zhong and Normey-Rico, 2002). Furthermore, Zhong and Mirkin quantitatively analyzed robust stability regions, some specifications of the disturbance response and the stability of the controller itself in IPDTs (Zhong and Mirkin, 2002). To reduce the long recovery time of the disturbance response, a dead-beat disturbance response is obtained while the robust stability is still guaranteed (Zhong, 2003a).

In industrial applications, the proportional-integral-derivative (PID) control is widely used. For time-delay systems, many corresponding methods to design PID controller have been developed. For IPDTs, a fine-tuned PI controller is enough

to achieve good performance. Rivera et al. (1986) applied the internal model control method to tune a PI controller so that the tuning formula is only related to the closed-loop time constant. Tyreus and Luyben (1992) showed that the closed-loop time constant in Rivera et al. (1986) must be chosen carefully to avoid a very oscillatory response, and then proposed an alternative approach by specifying a closed-loop damping coefficient (relating to the smallest closed-loop time constant). Luyben Luyben (1996) extended the method in Tyreus and Luyben (1992) to a PID controller and achieved a smaller closed-loop time constant for the same closed-loop damping coefficient. Poulin and Pomerleau (1999) obtained the PI settings for the maximum peak resonance specification according to the ultimate cycle information of the process. Wang and Cluett (1997) designed a PID controller in terms of the desired control signal trajectory. Visioli (2001) optimized PID settings by minimizing ISE, ISTE or ITSE with a genetic algorithm. Chidambaram and Sree (2003) proposed a simple coefficient-matching method to obtain similar results to Visioli (2001). However, the limits of what a PI controller can achieve is not clear.

Another important aspect of TDS is the control implementation. Almost all robust controllers of TDS incorporate a distributed-delay block which is a finite-impulse-response block. It is not easy to implement this block due to the requirement of internal stability. A common way is to replace the distributed delay by the sum of a series of discrete delays (Manitius and Olbrot, 1979; Watanabe and Ito, 1981; Palmor, 1996). It has been proved that the system stability can be guaranteed provided that the number of approximation steps is large enough (Zhong, 2004, 2005a). Furthermore, two rational implementations based on the $\delta$-operator and the extended bilinear transformations were proposed in Zhong (2005b). The next problem is to find the minimum number of approximation steps to guarantee the system stability.

It is worth pointing out that in general, "implementation" of a control system is about the practical application or is hardware-related. In this thesis, however, "implementation" represents not only the hardware but also the software realization (algorithm, simulation, etc) of control strategies.

### 1.1.2 Networked control systems

With the rapid development of network technology and infrastructure, the data transmission of the network becomes quick, secure and reliable. Consequently, control over a network, especially over the Internet, turns into a more and more popular topic in both academic and industrial areas. It is easy to realize remote control, distributed control and collaborative control using the network. NCSs have been widely applied to, for example, multimedia education system using Internet (Nemoto et al., 1997), Internet-based control engineering laboratory (Overstreet and Tzes, 1999), Web-based telerobots (Taylor and Dalton, 2000), haptics collaboration over the Internet (Hespanha et al., 2002a; Hikichi et al., 2002; Shirmohammadi and Woo, 2004), mobile sensor networks (Ogren et al., 2004), remote surgery (Meng et al., 2004), automated highway systems and unmanned aerial vehicles (Seiler and Sengupta, 2001, 2005), etc. Generally, there are two kinds of NCS configurations, direct structure networked control system and hierarchical structure networked control system (Yang et al., 2002a). The main difference between them is that there exists a local controller at the actuator side in the hierarchical structure. Compared to traditional control systems, however, NCS introduces some special problems. One obvious problem is the network-transmission delay. This delay might be constant, stochastic or completely random (Zhang et al., 2001; Montestruque and Antsaklis, 2004; Hespanha, 2006; Battilotti, 2008). From this point of view, NCS is a special case of TDS. Other problems in NCS include the network bandwidth limits, the network traffic congestion, the data-packet dropout and so on (Wong and Brockett, 1999; Elia and Mitter, 2001; Hespanha et al., 2002b; Nair and Evans, 2003; Tatikonda and Mitter, 2004). All these problems provide challenges for the analysis of system stability, the optimization of network scheduling, the design of control schemes and the system implementation.

Since stability is an important issue for the design and analysis of NCS, many methods have been proposed to cover this topic. The stability regions and hybrid systems technique were used to analyze the stability of the NCS by (Zhang et al., 2001) under the assumption that the network-induced delay is less than the sampling period. A perturbation method of stability analysis of NCS (Walsh et al.,

2001) was given for the case where the delay exists only in sensor-to-controller channel. A model-based method was reported in Montestruque and Antsaklis (2004). The sampling time scheduling method (Hong, 1995) was first proposed, which is used to appropriately select a sufficient long sampling such that the delay does not influence the control performance and the system can remain stable. However, this method is only applicable for the one dimension case. Park et al. (2002); Kim et al. (2003) extended it to multidimensional cases. The stability of NCS with data-packet disorder has been analyzed using the stochastic Lyapunov function and the jump linear system method in Zhang et al. (2001); Montestruque and Antsaklis (2003); Zhu et al. (2005); Liu et al. (2007a). Switched system theory is used to analyze the stability of NCS with random network delay in the feedback channel Liu et al. (2007b). However, duo to the delay randomness, the stability of the resulting system has to be subject to arbitrary switching, which makes the corresponding stability conditions quite restrict. So how to relax theses conditions deserves some study.

The design of control schemes in NCS is mainly subject to network-induced delay. Nilsson et al. (1998) used linear quadratic Gaussian (LQG) stochastic optimal method to design the controller which is independent of the network delay. A network delay dependent compensation method (Yoo et al., 2002) was developed using linear matrix inequality (LMI) based delay-dependent optimization. Adaptive neural control (Ge et al., 2003) was presented to compensate for the unknown network delays. (Liu et al., 2004) firstly proposed a networked predictive control (NPC) method for NCS. It includes a control prediction generator (CPG) at the controller side to generate a sequence of future controls. This sequence is fully sent to the actuator side. Then a network delay compensator (NDC) is designed at the actuator side to choose the appropriate element as the true control signal from the prediction sequence to compensate for the network delay. At this stage, in NPC method, the CPG is based on an original controller for the network-free system, but the resulting controller for the networked system is of form including a high order (nonlinear) term w.r.t. the original known controller. This makes its design difficult. On the other hand, the resulting controller also includes high order terms about the model information, so the robustness analysis is also a challenge.

Different to the traditional control systems, the implementation of NCSs is not trivial. For simulation implementation, a good description of the network is needed, which should represent the real network as much as possible. For real-time implementation, extra hardware costs are required for the information change between networks. Much work has been down for the network simulation, for example, NS2 (network simulation 2), `http://www.isi.edu/nsnam/ns/index.html`, TrueTime (Cervin et al., 2003) and NCS-sim, (`http://www.sussex.ac.uk/Users/taiyang/`). NetCon, `http://system.research.glam.ac.uk`, also provides an ideal hardware support for NCSs. For the sake of efficiency, it is necessary to make the implementation of the network-related parts of the system easy and quick. This can be realized by integrating into a toolbox.

## 1.2 Overview of this thesis

Chapter 2 of this thesis discusses various issues about integral processes with dead time (IPDT) controlled by a PI controller in the frequency domain. These issues include the control parameters region to stabilize the system, the control parameters to achieve the given gain and/or phase margins (GPM), the constraint on achievable gain and phase margins. These results are obtained on the basis of the normalized system.

Chapter 3 gives an estimation of the minimal number of implementation steps of distributed delays in linear control laws for time-delay systems is given. This is obtained by solving an inequality with respect to the number of implementation steps. A coarse estimation is given as the initial value to solve the inequality using bisection algorithms. A minimization process as well as some other techniques are also introduced to further improve the estimation. Examples show that the estimation provides a useful guide for the choice of the number of implementation steps.

Chapter 4 proposes an improved network-induced delay compensation strategy for networked predictive control systems. By deliberately increasing the delay purely induced by the network, we can partially assign the network-induced delay. The proposed method conveniently deals with the data-packet dropout and disorder in the

communication network. Such delay assignment is necessary when considering the system stability which addresses some limits on the network-induced delay. Two stability criteria are given for the closed-loop system with random network-induced delay, and a resulting implementation algorithm is also provided. A numerical example demonstrates the effectiveness of the presented method. A NPCS with random network-induced delay is discussed as a switched system. In the original delay-compensation method, the switching signal is not designable which results that all subsystems have to be stable and that they share a common Lyapunov function.

In Chapter 5, the control of magnetic levitation system over a network is considered. Firstly, a test-rig is built which allows the implementation of networked control. In order to improve the control performance, networked predictive method is employed, where the system model is identified online. A controller using LQG is designed based on the obtained model. Local control and networked control are both implemented in this test-rig, and also networked predictive control shows some advantages over the normal networked control which does not compensate for the network-induced delay.

Chapter 6 introduces the basic principle of a MATLAB/Simulink toolbox for networked control systems. It mainly includes control schemes, network simulation, network interface, and plant interface. This toolbox can be used for both simulation studies and also practical application of NCSs. Various control schemes are discussed and compared within the toolbox framework. Using the NCS toolbox, the implementation of NCSs can be achieved very quickly and users can focus on the study of new control schemes.

In Chapter 7, the major contributions of this thesis are summarized and some possible topics for future research are outlined.

Chapter 2 and 3 address problems in the time-invariant systems where the time delay is constant. Chapter 4, 5 and 6 discuss time-variant systems, of which NCS is a typical one where the network-induced delay is often time-varying. If the network-induced delay can be modelled as constant, then the results in Chapter 2 and 3 can be applied in some ways.

Some chapters in this thesis are based on one or more published or submitted

papers by the author. Chapter 2 is based on Wang et al. (2006). Chapter 3 is based on Wang et al. (2007c). Chapter 4 is based on Wang et al. (2007a,b,d). Chapters 5 and 6 are already applied to the NCS Lab on `http://www.ncslab.net`, which is developed by the author's research unit.

# Chapter 2

# PI Control of Integral Processes with Dead Time

This chapter discusses various issues about integral processes with dead time (IPDT) controlled by a PI controller. Firstly, the region of the control parameters to guarantee the system stability is characterized. Then, the control parameters to achieve the given gain and/or phase margins (GPM) are designed. Furthermore, the constraint on achievable gain and phase margins is derived. Two other system performance indexes, setpoint response ISE and disturbance response ISE, are also discussed quantitatively, which generally contradict to the system robustness. These results are obtained on the basis of the normalized system which involves only two free parameters. In order to simplify the tuning of the controller, a single tuning-parameter PI controller is designed according to the sub-ideal disturbance rejection. As a result, all conclusions about the above system performances are more restricted compared to what the typical PI can achieve. A 2-DOF structure is further proposed to improve the system by separating the setpoint response from the robustness and disturbance rejection.

## 2.1 Introduction

An integral process with dead time (IPDT) is a typical case of unstable processes. The classical Smith predictor (SP) is not applicable to IPDT because the integral mode will result in a steady-state error for a constant load disturbance. In the last two decades, many modified Smith predictor (MSP) and other schemes have been presented to overcome this shortcoming and to further improve the system performances. Watanabe and Ito (1981) proposed a process-model control to obtain the zero steady-state error and the desired transient response for the step disturbance. Åström et al. (1994) designed a structure to decouple the disturbance response and the setpoint response so that they can be designed separately. Normey-Rico and Camacho (1999) introduced a filter into the scheme of Watanabe and Ito (1981) to achieve the same setpoint and disturbance responses, while the robustness was taken into account explicitly. Zhong and co-authors published a series of papers devoted to the control of IPDT. In Zhong and Li (2002), a repetitive control scheme is used to improve the disturbance-rejection ability and to make the system possessing PID control advantages. In Zhong and Normey-Rico (2002), a disturbance-observer-based control scheme, which can reject arbitrary disturbances, was proposed and the resulting disturbance responses are in fact sub-ideal. In Zhong and Mirkin (2002), the robust stability regions, the disturbance response specifications and the stability of the controller itself were analyzed quantitatively. In Zhong (2003a), a dead-beat disturbance response was designed to reduce the long recovery time of the disturbance response, while the robust stability is still guaranteed. Lu et al. (2005) proposed a double two-degree-of-freedom (2-DOF) control scheme to improve the performance for both setpoint and disturbance responses.

Many PID-type control methods have also been proposed for IPDT. Rivera et al. (1986) applied the internal model control (IMC) method to tune a PI controller so that the tuning formula is only related to the closed-loop time constant. Tyreus and Luyben (1992) showed that this constant must be chosen carefully to avoid a very oscillatory response, and then proposed an alternative approach by specifying a closed-loop damping coefficient (relating to the smallest closed-loop time constant). Luyben (1996) extended the method in Tyreus and Luyben (1992) to a PID con-

troller and achieved a smaller closed-loop time constant for the same closed-loop damping coefficient. Poulin and Pomerleau (1999) obtained the PI settings for the maximum peak resonance specification according to the ultimate cycle information of the process. Wang and Cluett (1997) designed a PID controller in terms of the desired control signal trajectory. Visioli (2001) optimized the PID settings by minimizing ISE, ISTE or ITSE using a genetic algorithm. Chidambaram and Sree (2003) proposed a simple coefficient-matching method to obtain similar results to Visioli (2001). In the above-mentioned papers, however, it is still not clear how the control parameters affect the system stability and performances.

This chapter firstly characterizes the stability region of the control parameters. In this region, the effect of the control parameters on the system relative stability is further described quantitatively by using the stability margins. Consequently, the controller can be designed according to the specified gain and/or phase margins (GPM). It is also found that there exists a constraint on the achievable gain and phase margins (see Su et al. (2003) and the references therein for more details about fundamental performance limitations in control). These results are based on the normalized system which involves only two free parameters. A relevant paper about the stability region is Silva et al. (2002), where the complete set of stabilizing PID parameters is determined for both open-loop stable and unstable first order plus dead time processes (FOPDT), by applying Hermite-Biehler theorem to quasi-polynomials. The setpoint response and disturbance rejection performances are also discussed based on the normalized system. On the other hand, a single tuning-parameter PI controller is derived to simplify the controller tuning and the corresponding system performances are analyzed. A 2-DOF control, which is in fact a modified Smith predictor structure, is also provided to alleviate the tension among the system performances.

## 2.2 System normalization

Consider a classical feedback control system shown in Figure 2.1, where the process

$$G(s) = P(s)e^{-\tau s} \tag{2.1}$$

comprises an integrator part

$$P(s) = \frac{k}{s} \tag{2.2}$$

with $k > 0$ and a delay part $e^{-\tau s}$ with dead time $\tau > 0$; the controller

$$C(s) = k_p(1 + \frac{1}{T_i s}) \tag{2.3}$$

is a typical PI controller with the proportional gain $k_p > 0$ and the integral time constant $T_i > 0$; $r$, $y$, $d$, $u$, $e$ are the system input, output, disturbance, control and error, respectively.

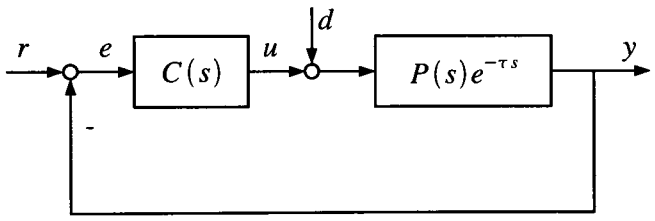



Figure 2.1: Classical feedback control system.

From Figure 2.1, the loop transfer function of the system is

$$\begin{aligned} L(s) &= C(s)G(s) \\ &= k_p(1 + \frac{1}{T_i s})\frac{k}{s}e^{-\tau s} \end{aligned} \tag{2.4}$$

which involves four parameters, i.e., $k$, $\tau$, $k_p$ and $T_i$. If the process parameters $k$ and $\tau$ are normalized into the following control parameters

$$\bar{k}_p = \tau k k_p \tag{2.5}$$

$$\bar{T}_i = \frac{T_i}{\tau}, \tag{2.6}$$

the loop transfer function (2.4) is correspondingly normalized as

$$\bar{L}(s) = \bar{k}_p(1 + \frac{1}{\bar{T}_i s})\frac{1}{s}e^{-s}, \tag{2.7}$$

which can be regarded as a system with the process

$$\bar{G}(s) = \frac{1}{s}e^{-s} \tag{2.8}$$

and the controller

$$\bar{C}(s) = \bar{k}_p(1 + \frac{1}{\bar{T}_i s}), \tag{2.9}$$

where $\bar{k}_p > 0$ and $\bar{T}_i > 0$ are the normalized proportional gain and the normalized integral time constant, respectively. The normalized system (2.8) and (2.9) involves only two free parameters $\bar{k}_p$ and $\bar{T}_i$, which, as can be seen later, considerably simplifies the system analysis and design. The results about the stability and the control performances of the normalized system can be easily converted to those of the original system, mainly based on the relationship (2.5) and (2.6).

## 2.3 Stability region

As a matter of fact, (2.7) can be obtained from (2.4) by substituting $s$ with $\frac{1}{\tau}s$, i.e.,

$$\begin{aligned} L(s) &= \tau k k_p(1 + \frac{1}{\frac{T_i}{\tau}\tau s})\frac{1}{\tau s}e^{-\tau s} \\ &= \bar{k}_p(1 + \frac{1}{\bar{T}_i \tau s})\frac{1}{\tau s}e^{-\tau s} \\ &= \bar{L}(\tau s). \end{aligned} \tag{2.10}$$

This means that the Nyquist plots of $L(s)$ and $\bar{L}(s)$ have the same form, but $L(s)$ reaches a point in the plot at a frequency $\tau$ times the frequency at which $\bar{L}(s)$ reaches the same point. Therefore, it is sufficient to analyze the stability of the normalized system to derive that of the original system.

Rewrite $\bar{L}(s)$ as $\bar{L}(j\omega) = \mathrm{Re}(\omega) + j\mathrm{Im}(\omega)$, where

$$\mathrm{Re}(\omega) = -\frac{\bar{k}_p \cos\omega}{\bar{T}_i\omega^2} - \frac{\bar{k}_p \sin\omega}{\omega}$$

$$\mathrm{Im}(\omega) = \frac{\bar{k}_p \sin\omega}{\bar{T}_i\omega^2} - \frac{\bar{k}_p \cos\omega}{\omega}.$$

When $\omega \to 0$, the real part satisfies

$$\lim_{\omega\to 0} \mathrm{Re}(\omega) = -\infty$$

and the imaginary part has the following three cases

$$\lim_{\omega\to 0} \mathrm{Im}(\omega) = \begin{cases} -\infty & \bar{T}_i > 1 \\ 0 & \bar{T}_i = 1 \\ +\infty & 0 < \bar{T}_i < 1 \end{cases}.$$

The Nyquist plots for these cases are shown in Figure 2.2. Because $\bar{L}(s)$ has no poles on the right half complex plane (RHP), the Nyquist plots should not encircle the point $(-1, 0)$ to guarantee the closed-loop system stability. In the case of $0 < \bar{T}_i < 1$ or $\bar{T}_i = 1$, the Nyquist curve starts above the real axis and will definitely encircle the point $(-1, 0)$. In the case of $\bar{T}_i > 1$, it is possible for the Nyquist curve not to encircle the point $(-1, 0)$. The above analysis results in the following necessary condition for the closed-loop system stability.

**Theorem 2.1** *The closed-loop feedback system with the open-loop transfer function given in (2.7) is stable only if* $\bar{T}_i > 1$.

According to (2.6), Theorem 2.1 means that in the original system the integral time constant of the PI controller $C(s)$ must be greater than the dead time of the process $G(s)$. Furthermore, a necessary and sufficient condition for the stability is given below.

**Theorem 2.2** *The closed-loop feedback system with the open-loop transfer function given in (2.7) is stable if and only if* $\bar{k}_p$ *and* $\bar{T}_i$ *satisfy*

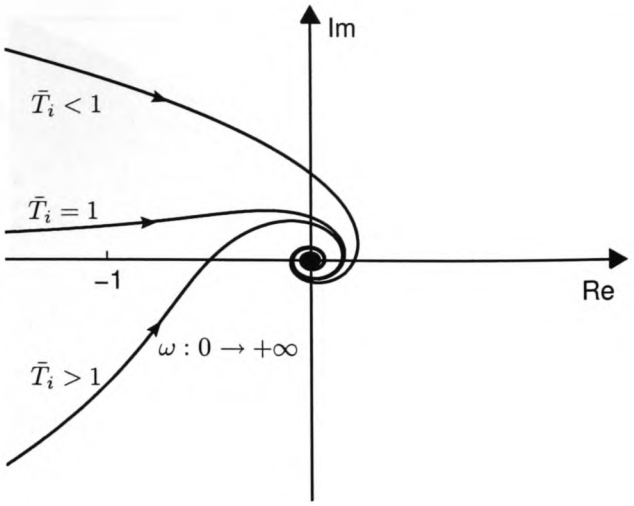



Figure 2.2: Nyquist plot of the loop transfer function $\bar{L}(s)$.

$$\bar{k}_p < \frac{\bar{T}_i \omega_p^2}{\sqrt{\bar{T}_i^2 \omega_p^2 + 1}}, \tag{2.11}$$

*where* $\omega_p$ *is the solution of*

$$\omega_p = \arctan(\bar{T}_i \omega_p) \tag{2.12}$$

*in the interval* $(0, \frac{\pi}{2}]$.

**Proof**: The closed-loop feedback system is stable if and only if the Nyquist curve crosses the real axis from the right side of the point $(-1, 0)$, i.e., satisfies $\mathrm{Re}(\omega) > -1$ when $\mathrm{Im}(\omega) = 0$. This gives

$$\frac{\bar{k}_p \cos \omega_p}{\bar{T}_i \omega_p^2} + \frac{\bar{k}_p \sin \omega_p}{\omega_p} < 1 \tag{2.13}$$

$$\frac{\sin \omega_p}{\bar{T}_i \omega_p} - \cos \omega_p = 0, \tag{2.14}$$

where $\omega_p$ is called the phase crossover frequency. Because $\bar{T}_i > 1$, the minimum solution of $\omega_p$ in (2.14) lies in the interval $(0, \frac{\pi}{2}]$, where $\frac{\pi}{2}$ is obtained when $\bar{T}_i \to +\infty$ (in this situation, the PI controller (2.9) degenerates to a P-type controller). This

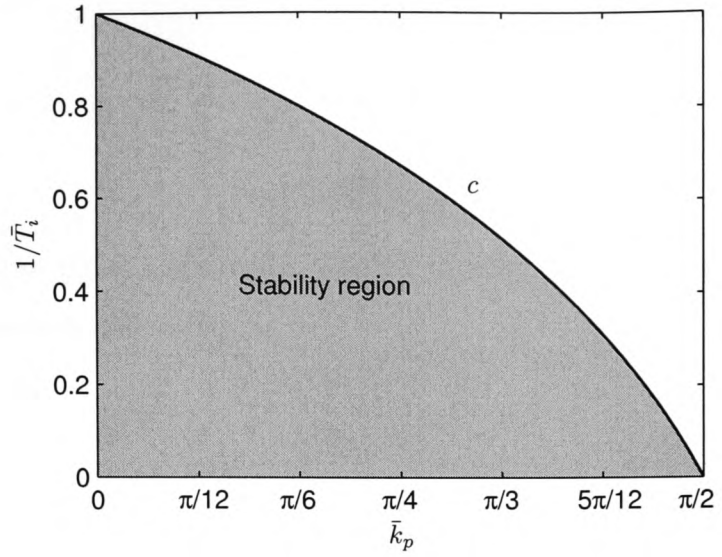



Figure 2.3: Stability region of the control parameters $\bar{k}_p$ and $\bar{T}_i$.

means that the Nyquist curve crosses the real axis for the first time at a frequency not more than $\frac{\pi}{2}$. Thus, (2.14) can be converted into (2.12). Furthermore, simplifying (2.13) with (2.14) gives (2.11). This completes the proof. □

When the Nyquist curve crosses the point $(-1, 0)$, there is

$$\bar{k}_p = \frac{\bar{T}_i \omega_p^2}{\sqrt{\bar{T}_i^2 \omega_p^2 + 1}}. \tag{2.15}$$

From this, together with (2.12), the relationship between $\bar{k}_p$ and $\bar{T}_i$ can be solved numerically, which is shown in Figure 2.3 as the curve $c$. Note that when $\bar{T}_i \to +\infty$, $\omega_p \to \frac{\pi}{2}$ and $\bar{k}_p \to \frac{\pi}{2}$; when $\bar{k}_p \to 0$, $\omega_p \to 0$ and $\bar{T}_i \to 1$. Clearly, the filled area in Figure 2.3 corresponds to (2.11). This is the stability region.

## 2.4 Robustness

The robustness of the system describes its capacity to maintain stable when the process has parameter uncertainties or unmodelled dynamics. The robustness can be quantitatively expressed by the stability margins, such as gain margin $A_m$ or phase margin $\phi_m$. They are defined as

$$A_m = \frac{1}{|\bar{L}(j\omega_p)|} \tag{2.16}$$

$$\phi_m = \arg[\bar{L}(j\omega_g)] + \pi, \tag{2.17}$$

where $\omega_p$ is the phase crossover frequency defined in (2.12) and $\omega_g$ is the gain crossover frequency defined by

$$|\bar{L}(j\omega_g)| = 1. \tag{2.18}$$

Conventionally, the gain margin, which is larger than 1, is converted to have the unit "dB". Here it is kept dimensionless to simplify the expression and calculation. Before the Nyquist curve reaches the real axis for the first time, $\omega \leq \frac{\pi}{2}$ and $\mathrm{Re}(\omega)$ is always negative, so $\phi_m$ is in the range of $0 < \phi_m < \frac{\pi}{2}$.

Substituting (2.7) into (2.16) and (2.17) gives

$$A_m = \frac{\bar{T}_i\omega_p^2}{\bar{k}_p\sqrt{\bar{T}_i^2\omega_p^2 + 1}} \tag{2.19}$$

$$\phi_m = \arctan(\bar{T}_i\omega_g) - \omega_g, \tag{2.20}$$

where $\omega_p$ is given in (2.12) and $\omega_g$, according to (2.18), is

$$\omega_g = \frac{\sqrt{2}}{2}\sqrt{\bar{k}_p^2 + \sqrt{\bar{k}_p^4 + \frac{4\bar{k}_p^2}{\bar{T}_i^2}}}. \tag{2.21}$$

For a given gain margin $A_m$, the parameters $\bar{k}_p$ and $\bar{T}_i$ satisfying (2.19) and (2.12) can be solved numerically. The solutions for typical $A_m = 2, 3, 4, 5, 6$ are shown in Figure 2.4 and are called gain-margin curves. When $\bar{k}_p \to 0$, $\bar{T}_i \to 1$. This is the point $(0, 1)$. When $\bar{T}_i \to +\infty$, $\bar{k}_p \to \bar{k}_p^{(A)}$ with

$$\bar{k}_p^{(A)} = \frac{\pi}{2A_m}. \tag{2.22}$$

This is the intersection point of the gain-margin curve and the horizontal axis.

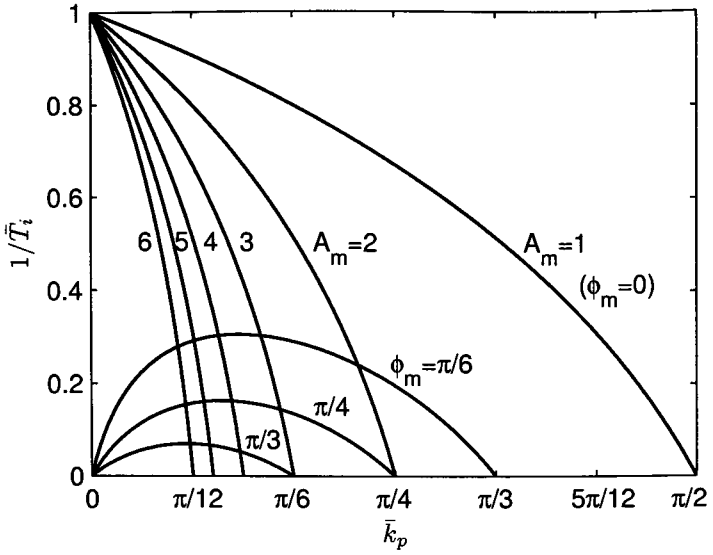



Figure 2.4: Typical stability margins $A_m$ and $\phi_m$.

Similarly, for a given phase margin $\phi_m$, the parameters $\bar{k}_p$ and $\bar{T}_i$ satisfying (2.20) and (2.21) can also be solved numerically. The solutions for typical $\phi_m = \frac{\pi}{6}, \frac{\pi}{4}, \frac{\pi}{3}$ are also shown in Figure 2.4 and are called phase-margin curves. When $\bar{T}_i \to +\infty$, $\bar{k}_p \to 0$ or $\bar{k}_p \to \bar{k}_p^{(\phi)}$ with

$$\bar{k}_p^{(\phi)} = \frac{\pi}{2} - \phi_m. \tag{2.23}$$

This is the right-side endpoint of the phase-margin curve. Note that the curve for $A_m = 1$ or $\phi_m = 0$ is the curve $c$ shown in Figure 2.3.

So far, the relationship between the system gain margin or phase margin and the control parameters has been analyzed. It is easy to design $\bar{C}(s)$ from the specified margins, i.e., the intersection point of the relevant gain-margin curve and phase-margin curve in Figure 2.4. However, no arbitrary $A_m$ and $\phi_m$ can be achieved simultaneously. There is a constraint on the achievable stability margins.

**Theorem 2.3** *The achievable stability margins $A_m$ and $\phi_m$ satisfy the following constraint:*

$$\phi_m \leq \frac{\pi}{2}(1 - \frac{1}{A_m}).$$

**Proof**: For any pair $(A_m, \phi_m)$, there exists either an unique pair $(\bar{k}_p, \bar{T}_i)$ or no pair

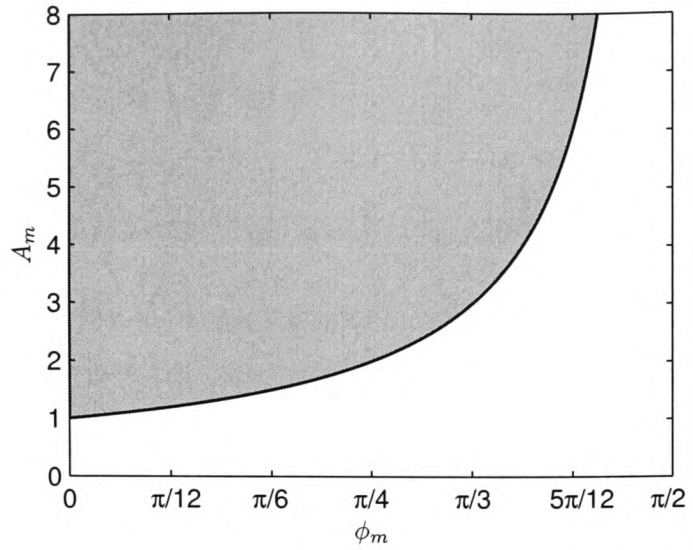



Figure 2.5: Achievable stability margins $A_m$ and $\phi_m$.

$(\bar{k}_p, \bar{T}_i)$ to meet them, depending on whether or not the corresponding gain-margin and phase-margin curves intersect in Figure 2.4. They intersect with each other when $\bar{k}_p^{(A)} \leq \bar{k}_p^{(\phi)}$. According to (2.22) and (2.23), it means

$$\frac{\pi}{2A_m} \leq \frac{\pi}{2} - \phi_m,$$

where the "=" is satisfied when $\bar{T}_i \to +\infty$. This gives the condition in the theorem. This completes the proof. □

The achievable stability margins are shown in the filled area in Figure 2.5, where the curve corresponds to the "=" case in the constraint in Theorem 2.3. When $A_m$ approaches $+\infty$, $\phi_m$ approaches $\frac{\pi}{2}$.

When the system is designed for the specified stability margins $A_m$ and $\phi_m$, the allowable uncertainties of the process parameters are determined too. Assuming that there exists an additive uncertainty $\Delta k$ in the process gain $k$ $(\Delta k + k > 0)$, according to (2.5) and the definition of the gain margin, the system is robustly stable if

$$-1 < \frac{\Delta k}{k} < A_m - 1. \tag{2.24}$$

Similarly, if there exists an additive uncertainty $\Delta\tau$ in the process dead time $\tau$

$(\Delta\tau + \tau \geq 0)$, the system is robustly stable when

$$-1 \leq \frac{\Delta\tau}{\tau} < \frac{\phi_m}{\omega_g}. \tag{2.25}$$

## 2.5 Setpoint tracking and disturbance rejection

There are many indexes to assess the system performance in tracking the setpoint and rejecting the disturbance. Among them, the integral square error (ISE) index

$$\begin{aligned} J &= \int_0^\infty e(t)^2 \mathrm{d}t \\ &= \int_0^\infty (r(t) - y(t, \bar{k}_p, \bar{T}_i))^2 \mathrm{d}t \end{aligned} \tag{2.26}$$

is widely used. This requires solving the following optimization problem.

**Problem 2.1**

$$\min_{\bar{k}_p, \bar{T}_i} J$$

*s.t.* $\bar{k}_p$ *and* $\bar{T}_i$ *are within the stability region.*

### 2.5.1 ISE for setpoint response

From Figure 2.1, the normalized setpoint response is

$$\bar{y}^{(r)}(s) = \frac{\bar{L}(s)}{1 + \bar{L}(s)} r(s).$$

Because the integral action in the controller, the steady-state of $\bar{y}^{(r)}(t)$ is $\bar{y}^{(r)}(\infty) = r_0$ for any step signal $r(t) = r_0 \cdot 1(t)$. As a result, the normalized setpoint ISE $\bar{J}^{(r)}$ is finite, and then the solution of Problem 2.1 is feasible. Due to the dead time term in the denominator in $\bar{y}^{(r)}(s)$, however, it is difficult to obtain the analytic expression of $\bar{y}^{(r)}(t)$ from $\bar{y}^{(r)}(t) = \mathcal{L}^{-1}(\bar{y}^{(r)}(s))$, where $\mathcal{L}^{-1}$ means the inverse Laplace transform. Thus a numerical calculation method has to be employed to solve Problem 2.1. Here, MATLAB/Simulink tool is used to simulate the system and calculate $\bar{y}^{(r)}(t)$. Considering $r(t)$ as a unit step signal, the system reaches its minimization of the

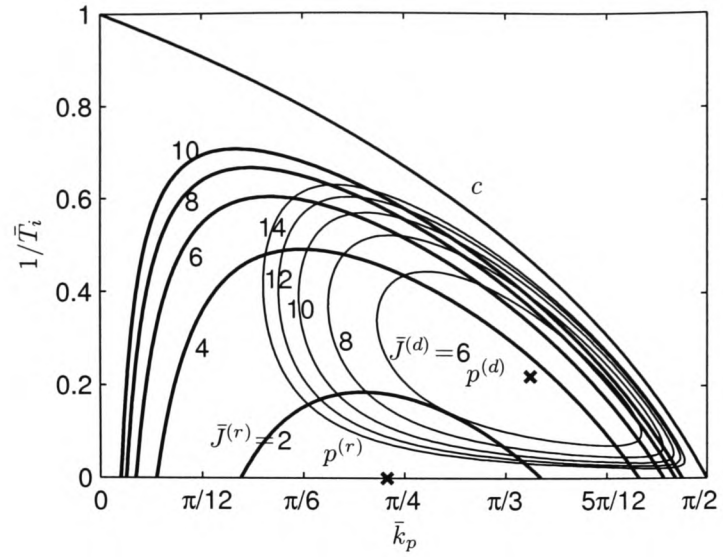



Figure 2.6: ISE of the setpoint and disturbance responses.

normalized setpoint ISE

$$\bar{J}^{(r)}_{\min} = 1.53$$

with the control parameters

$$\bar{k}_p^{(r)} = 0.74$$

$$\bar{T}_i^{(r)} = +\infty.$$

See point $p^{(r)}$ in Figure 2.6. That is to say, a P-type controller is setpoint-ISE optimal for the IPDT process. This can be attributed to the fact that the integral action is already included in the plant model, so any extra integral part in the controller might cause the system more oscillation. $\bar{J}^{(r)} = 2, 4, 6, 8, 10$ are also shown in Figure 2.6.

Note that the above analysis is based on the setpoint response of the normalized system. As far as the original system is concerned, there is the following theorem.

**Theorem 2.4** *For the unit step setpoint response, both the original system and the normalized system reach their minimizations of ISE at* $\bar{k}_p^{(r)}$ *and* $\bar{T}_i^{(r)}$*; Moreover, for any given* $\bar{k}_p$ *and* $\bar{T}_i$*, ISE of the original system is* $\tau$ *times that of the normalized system.*

**Proof**: From the relationship (2.10) about $L(s)$ and $\bar{L}(s)$, there is

$$\begin{aligned} y^{(r)}(s) &= \frac{L(s)}{1+L(s)}\frac{1}{s} \\ &= \tau \bar{y}^{(r)}(\tau s). \end{aligned}$$

According to the property of inverse-Laplace transform, the time-domain counterpart satisfies

$$y^{(r)}(t) = \bar{y}^{(r)}(\frac{1}{\tau}t).$$

Furthermore, the normalized ISE and the original ISE obey

$$\begin{aligned} J^{(r)}(\bar{k}_p, \bar{T}_i) &= \int_0^\infty (1 - y(t, \bar{k}_p, \bar{T}_i))^2 \mathrm{d}t \\ &= \int_0^\infty (1 - \bar{y}(\frac{1}{\tau}t, \bar{k}_p, \bar{T}_i))^2 \mathrm{d}t \\ &= \tau \int_0^\infty (1 - \bar{y}(t, \bar{k}_p, \bar{T}_i))^2 \mathrm{d}t \\ &= \tau \bar{J}^{(r)}(\bar{k}_p, \bar{T}_i). \end{aligned}$$

It is clear that $J^{(r)}$ and $\bar{J}^{(r)}$ are simply proportional and therefore they reach their minimizations at the same $\bar{k}_p$ and $\bar{T}_i$. This completes the proof. □

### 2.5.2 ISE for disturbance response

The normalized disturbance response is

$$\bar{y}^{(d)}(s) = \frac{\bar{G}(s)}{1+\bar{L}(s)} d(s).$$

Similar to the setpoint response, the steady-state $\bar{y}^{(d)}(\infty) = 0$ for any step disturbance $d(t) = d_0 \cdot 1(t)$. Therefore, the normalized disturbance $\bar{J}^{(d)}$ is finite and then Problem 2.1 is solvable (in this case, $r(t)$ is assumed 0). It is worth noting that a P-type controller results in $\bar{y}^{(d)}(\infty) = \frac{d_0}{\bar{k}_p}$. That is to say, the disturbance can not be rejected. This also implies that the system can not acquire the optimizations of the setpoint ISE and the disturbance ISE at the same time. For a unit step disturbance

$d$, point $p^{(d)}$ in Figure 2.6 approximately corresponds to

$$\bar{k}_p^{(d)} = 1.11$$

$$\bar{T}_i^{(d)} = 4.55$$

at which the system reaches the minimization of the normalized disturbance ISE

$$\bar{J}_{\min}^{(d)} = 4.05.$$

Figure 2.6 also gives the solution of control parameters for $\bar{J}^{(d)} = 6, 8, 10, 12, 14$.

**Corollary 2.1** *For the unit step disturbance response, both the original system and the normalized system reach their minimizations of ISE at the same* $\bar{k}_p^{(d)}$ *and* $\bar{T}_i^{(d)}$*; Moreover, for any given* $\bar{k}_p$ *and* $\bar{T}_i$*, ISE of the original system is* $\tau^3 k^2$ *times that of the normalized system.*

From Figure 2.6, it can be seen that a big $\frac{1}{\bar{T}_i}$ makes the setpoint-ISE sensitive; so does a small $\bar{k}_p$ or a big $\bar{k}_p$. For the disturbance ISE, there is a similar result. Moreover, good ISE indexes for the setpoint response and the disturbance response are generally contradictory, or otherwise both of them are unacceptably high. For example, in the neighborhood of $p^{(r)}$, the setpoint ISE is small, but the disturbance ISE is very large. Therefore, there has to be a compromise in the selection of $\bar{k}_p$ and $\bar{T}_i$ between these two performances. Considering the system robustness from Figure 2.4 again, the same conclusion is reached as far as selecting $\bar{k}_p$ and $\bar{T}_i$ values. It is difficult to obtain a set of control parameters to simultaneously satisfy all requirements.

## 2.6 PI controller with single tuning-parameter

In the above sections, control parameters $k_p$ and $T_i$ of the PI controller $C(s)$ need to be tuned in unison to achieve an acceptable comprise for different performances. This is not easy. In order to get a simple tuning rule, the PI controller is often converted to a type which has a single tuning-parameter.

It is known that the sub-ideal disturbance rejection for processes with dead time requires the system to have a disturbance transfer function of the form (Zhong and Mirkin, 2002)

$$G_d^{(sub)}(s) = (1 - Q(s)e^{-\tau s})P(s)e^{-\tau s}. \tag{2.27}$$

For the step disturbance, $Q(s)$ needs to meet

$$Q(0) = 1$$

$$\dot{Q}(0) = \tau$$

to make the system have zero steady-state error. It is clear that

$$Q(s) = \frac{(2\alpha + \tau)s + 1}{(\alpha s + 1)^2} \tag{2.28}$$

satisfies the above requirements, where $\alpha > 0$.

From Figure 2.1, the actual disturbance transfer function is

$$G_d(s) = \frac{P(s)e^{-\tau s}}{1 + C(s)P(s)e^{-\tau s}}. \tag{2.29}$$

Comparing (2.29) and (2.27) yields

$$C^{(sub)}(s) = \frac{Q(s)}{(1 - Q(s)e^{-\tau s})P(s)}. \tag{2.30}$$

Take the approximation $e^{-\tau s} \approx 1 - \tau s$ in (2.30) and substitute $P(s)$ and $Q(s)$ with (2.2) and (2.28), which gives

$$\begin{aligned} C^{(\alpha)}(s) &\approx \frac{Q(s)}{(1 - Q(s)(1 - \tau s))P(s)} \\ &= \frac{1}{k}\frac{(2\alpha + \tau)s + 1}{(\alpha + \tau)^2 s}. \end{aligned}$$

This is a PI controller with

$$k_p = \frac{1}{k}\frac{2\alpha + \tau}{(\alpha + \tau)^2}$$

$$T_i = 2\alpha + \tau.$$

Here $\alpha$ is the single tuning-parameter. $C^{(\alpha)}(s)$ can also be normalized as $\bar{C}^{(\bar{\alpha})}(s)$ with

$$\bar{k}_p = \frac{2\bar{\alpha}+1}{(\bar{\alpha}+1)^2} \tag{2.31}$$

$$\bar{T}_i = 2\bar{\alpha}+1, \tag{2.32}$$

where $\bar{\alpha} = \frac{\alpha}{\tau}$.

It is apparent that the stability region of $\bar{C}^{(\bar{\alpha})}(s)$, compared to $\bar{C}(s)$, is more restricted. Actually, this region degrades to a curve in the $\bar{k}_p - \bar{T}_i$ coordinate. Cancelling $\bar{\alpha}$ in (2.31) and (2.32) gives

$$\bar{k}_p = \frac{4\bar{T}_i}{(\bar{T}_i+1)^2}.$$

This is the curve $c^{(\bar{\alpha})}$ in Figure 2.7. The original point corresponds to $\bar{\alpha} = +\infty$. The intersection $p^{(c,\bar{c}^{(\bar{\alpha})})}$ of curve $c$ and $c^{(\bar{\alpha})}$ can be solved from equation (2.12), (2.15), (2.31) and (2.32) as

$$\bar{k}_p^{(c,\bar{c}^{(\bar{\alpha})})} = 0.93$$

$$\bar{T}_i^{(c,\bar{c}^{(\bar{\alpha})})} = 1.7$$

$$\bar{\alpha}^{(c,\bar{c}^{(\bar{\alpha})})} = 0.35.$$

From point $p^{(c,\bar{c}^{(\bar{\alpha})})}$ to the original point, $\bar{\alpha}$ is monotonically increasing. Therefore,

$$\bar{\alpha} > \bar{\alpha}^{(c,\bar{c}^{(\bar{\alpha})})}$$

is the condition for $\bar{C}^{(\bar{\alpha})}(s)$ to make the system stable. Correspondingly, the achievable control performance under the single parameter PI controller $\bar{C}^{(\bar{\alpha})}(s)$ is also restricted. Figure 2.8 shows the achievable stability margins. This is obtained by solving (2.12), (2.19), (2.20), (2.21), (2.31) and (2.32). The bigger the $\bar{\alpha}$, the more robust the system. The setpoint ISE and the disturbance ISE for $\bar{C}^{(\bar{\alpha})}(s)$ are shown

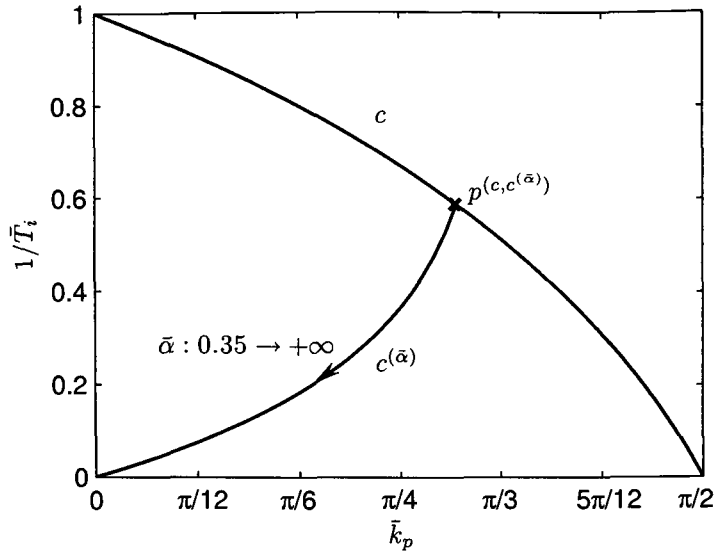



Figure 2.7: Stability region of the single tuning-parameter PI controller.

in Figure 2.9. It can be seen that their minimizations

$$\bar{J}_{\min}^{(r,\bar{\alpha})} = 2.09$$

$$\bar{J}_{\min}^{(d,\bar{\alpha})} = 5.37$$

are achieved at

$$\bar{\alpha}^{(r,\bar{\alpha})} = 2.34$$

$$\bar{\alpha}^{(d,\bar{\alpha})} = 0.82,$$

respectively. They are bigger than $\bar{J}_{\min}^{(r)}$ and $\bar{J}_{\min}^{(d)}$ corresponding to $\bar{C}(s)$. Furthermore, in $0.35 < \bar{\alpha} < 2.34$ or $0.35 < \bar{\alpha} < 0.82$, the bigger the $\bar{\alpha}$, the better the setpoint tracking or the disturbance rejection; in $\bar{\alpha} > 2.34$ or $\bar{\alpha} > 0.82$, the bigger the $\bar{\alpha}$, the worse the setpoint response or the disturbance response. Because in $0.35 < \bar{\alpha} < 0.82$, $\bar{J}^{(d,\bar{\alpha})}$ is very sensitive, it is better to take $\bar{\alpha} > 0.82$. Thus, the impact of $\bar{\alpha}$ on the system performances (setpoint tracking, disturbance rejection and robustness) is quite clear. The design of $\bar{\alpha}$ is straightforward by compromising these three performances.

The optimal ISE indexes for the setpoint response and the disturbance response

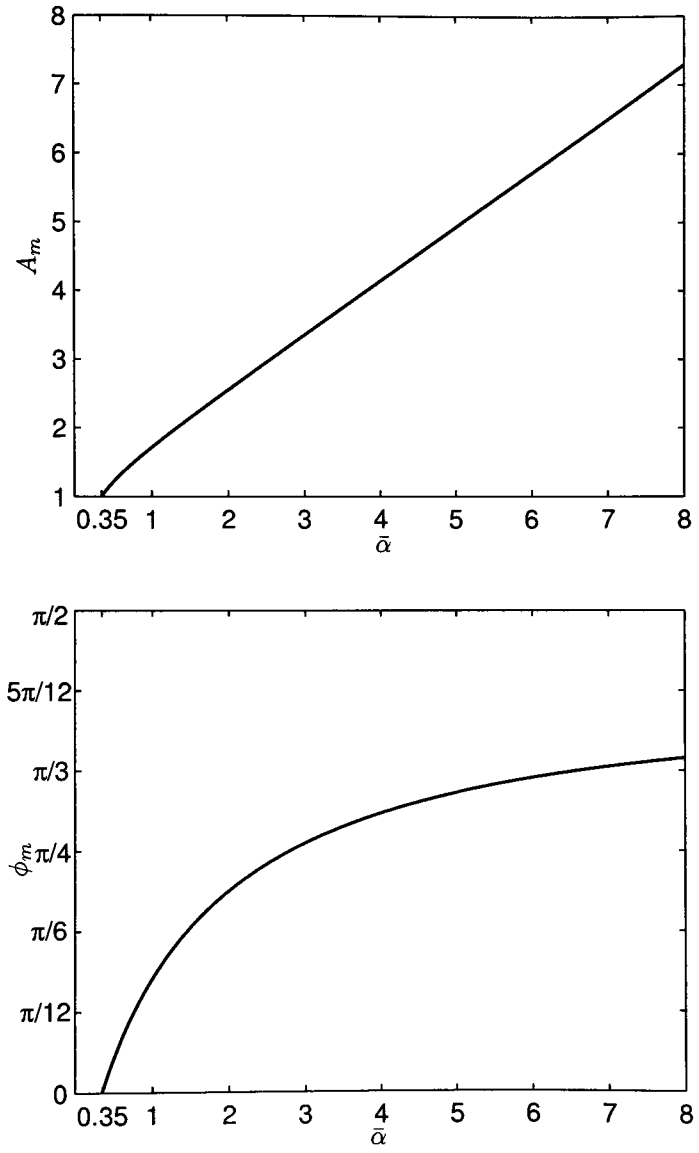



Figure 2.8: Achievable stability margins $A_m$ and $\phi_m$ of the single tuning-parameter PI controller.

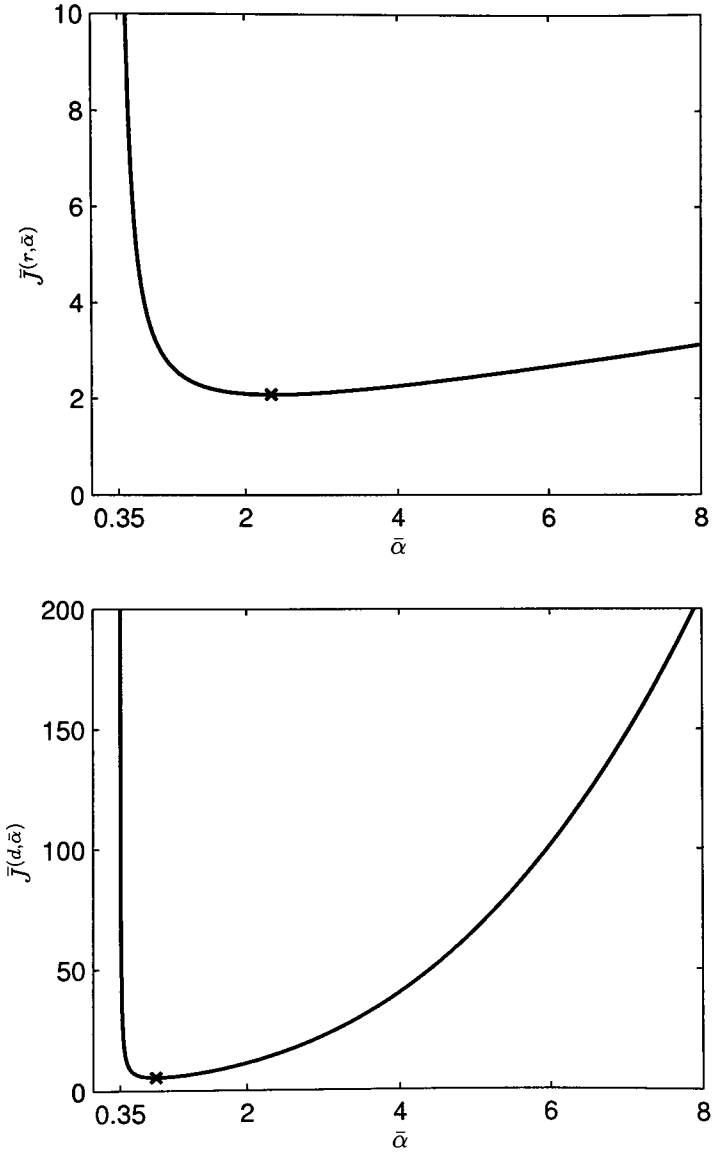



Figure 2.9: ISE of the setpoint and disturbance responses of the single tuning-parameter PI controller.

Table 2.1: Optimal ISE index for unit step signal.

| | | Optimal ISE | $k_p$ | $T_i$ | $\alpha$ |
|---|---|---|---|---|---|
| Typical PI Controller $C(s)$ | Setpoint response | $1.53\tau$ | $\frac{0.74}{\tau k}$ | $\infty$ | - |
| | Disturbance response | $4.05\tau^3k^2$ | $\frac{1.11}{\tau k}$ | $4.55\tau$ | - |
| Single tuning-parameter PI Controller $C^{(\alpha)}(s)$ | Setpoint response | $2.09\tau$ | $\frac{0.51}{\tau k}$ | $5.68\tau$ | $2.34\tau$ |
| | Disturbance response | $5.37\tau^3k^2$ | $\frac{0.8}{\tau k}$ | $2.64\tau$ | $0.82\tau$ |

of the original system with typical or single tuning-parameter PI controller are summarized in Table 2.1.

**Remark 2.1** *It should be pointed out that essentially $C(s)$ can achieve a more satisfactory control performance than $C^{(\alpha)}(s)$, provided that $C(s)$ is carefully tuned. What $C^{(\alpha)}(s)$ can achieve is only a subset of what $C(s)$ can achieve. The advantage of $C^{(\alpha)}(s)$ lies in its simple tuning rule.*

## 2.7 2-DOF control scheme

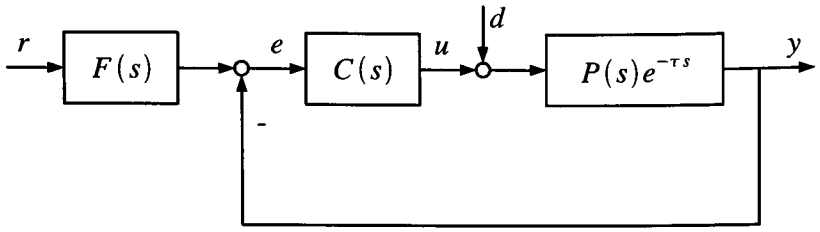
Due to the performance limits that the classical control system is subject to, a 2-DOF control scheme is proposed to improve different control performances individually; see Figure 2.10(a). Its main idea is to separate the setpoint response from the disturbance response and the robustness. In more detail, the first control degree of freedom $F(s)$ is for the setpoint response; the second control degree of freedom $C(s)$ only needs to compromise between the disturbance rejection and the robustness.

$F(s)$ is designed using direct synthesis from the desired setpoint response. In Figure 2.10(a), the transfer function from the setpoint $r$ to the output $y$ is

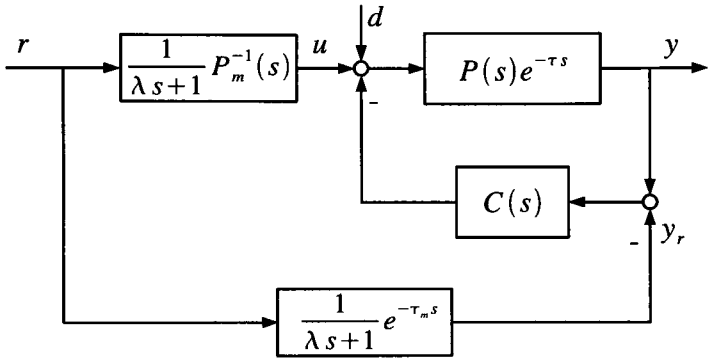
$$G_r(s) = F(s)\frac{C(s)P(s)e^{-\tau s}}{1 + C(s)P(s)e^{-\tau s}}.$$

$F(s)$ is designed as

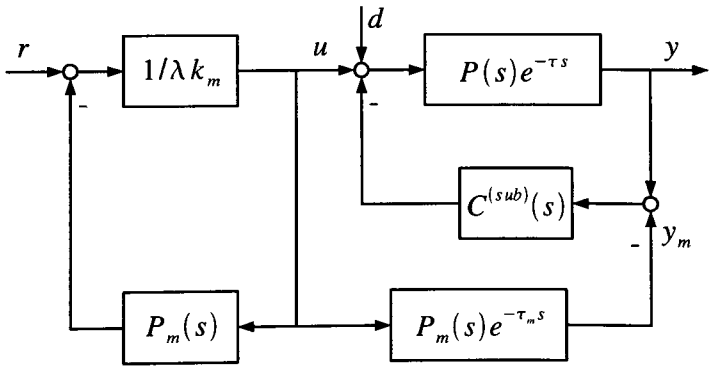
$$F(s) = \frac{1 + C(s)P_m(s)e^{-\tau_m s}}{C(s)P_m(s)} \cdot \frac{1}{\lambda s + 1},$$




(a) Design structure




(b) Implementation structure




(c) Modified Smith predictor structure

Figure 2.10: Two-degree-of-freedom control structure for integral processes with dead time.

where $P_m(s) = \frac{k_m}{s}$ and $e^{-\tau_m s}$ denote the nominal model of processes $P(s)$ and $e^{-\tau s}$, respectively. If there is no model-process mismatching, i.e., $P_m(s) = P(s)$ and $e^{-\tau_m s} = e^{-\tau s}$, the desired setpoint response is

$$G_r(s) = \frac{1}{\lambda s + 1} e^{-\tau s}.$$

This is independent of the second degree of freedom $C(s)$. Here, $\frac{1}{\lambda s+1}$ $(\lambda > 0)$ is selected to make $F(s)$ proper. To tune $\lambda$ is to tune the response speed. Note that there is no unstable zero-pole cancellation between $F(s)$ and the feedback loop because the closed-loop system is designed to be stable. On the other hand, the transfer function from the disturbance $d$ to the output $y$, same as (2.29), is independent of the first degree of freedom $F(s)$. Therefore, the setpoint response and the disturbance response are decoupled from each other.

Rewrite $F(s)$ as

$$F(s) = \frac{C(s)^{-1} P_m(s)^{-1}}{\lambda s + 1} + \frac{1}{\lambda s + 1} e^{-\tau_m s}.$$

This results in an equivalent structure as shown in Figure 2.10(b). It is useful for implementation because $F(s)$ in Figure 2.10(a) is dependent on the parameters of $C(s)$. Moreover, from this structure, it is clear that the desired setpoint response is separated out as the reference output $y_r$. $C(s)$ acts only when the error between the actual output $y$ and the reference output $y_r$ is not equal to 0. If there are no disturbances or process uncertainties in the system, $y$ is always equal to $y_r$ and $C(s)$ has no effect on the system, i.e., the system becomes open-loop.

The second control degree of freedom $C(s)$ can take the typical or single tuning-parameter PI-type. Thus, all results about the disturbance response and the robustness discussed in the previous sections can be applied here directly. Note that if $C(s)$ takes $C^{(sub)}(s)$ in (2.30), Figure 2.10(b) can be further converted to a modified Smith predictor structure as Figure 2.10(c), where $y_m$ is the nominal output.

Table 2.2: Typical PI controller parameters for Example 1.

| | $\bar{k}_p$ | $\frac{1}{T_i}$ | $k_p$ | $T_i$ | $J^{(r)}$ | $J^{(d)}$ | $A_m$ | $\phi_m$ |
|---|---|---|---|---|---|---|---|---|
| Case 1: optimal disturbance ISE | 1.11 | 0.22 | 3.66 | 27.27 | 20.16 | 2.24 | 1.26 | $\frac{\pi}{12.7}$ |
| Case 2: optimal setpoint ISE | 0.74 | 0 | 2.44 | $+\infty$ | 9.18 | $+\infty$ | 2.12 | $\frac{\pi}{3.78}$ |
| Case 3: big stability margins | 0.49 | 0.14 | 1.61 | 42.86 | 12.08 | 10.12 | 3 | $\frac{\pi}{4}$ |
| Case 4: compromise | 0.53 | 0.29 | 1.75 | 20.69 | 14.83 | 5.52 | 2.5 | $\frac{\pi}{6}$ |

## 2.8 Examples

### 2.8.1 Example 1: classical feedback control

Firstly, the classical control structure and typical PI controller are studied in this example. In Figure 2.1, consider a widely studied process (Wang and Cluett, 1997; Visioli, 2001; Chidambaram and Sree, 2003)

$$G(s) = \frac{0.0506}{s} e^{-6s}.$$

The input $r(t)$ and the disturbance $d(t)$ are both unit step signals. The disturbance is added into the system at time 150 seconds. Four sets of control parameters for the PI controller $C(s)$ are compared in Table 2.2.

The nominal system response is shown in Figure 2.11. Case 1 gives the best disturbance rejection, but the worst setpoint tracking. Case 2 gives the best setpoint tracking, but its disturbance response is worst. Actually, Case 2 can not reject the disturbance because the resulting controller is a P-type. Case 3 and Case 4 give a compromise between these two performances. In more detail, Case 3, compared to Case 4, has a better setpoint response but a worse disturbance response. The simulation results correspond to the numerical calculations in Table 2.2 and verify the conclusions.

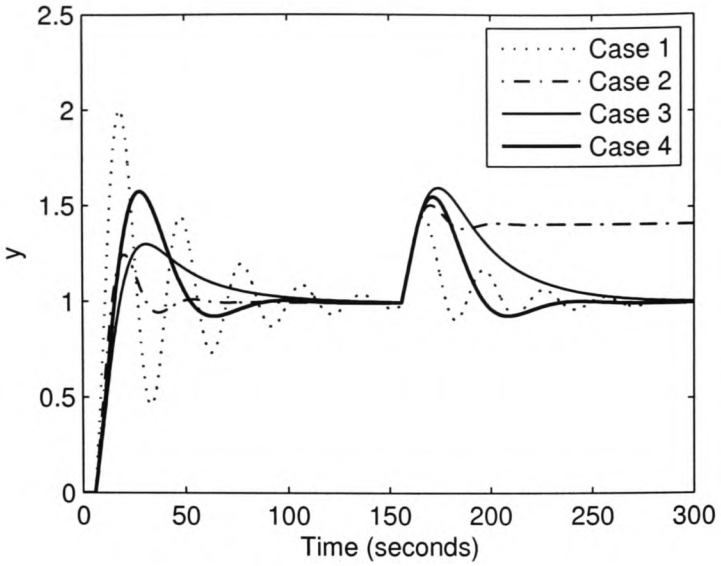



Figure 2.11: Nominal responses.

When the process has a 30% uncertainty in its gain, i.e., $\Delta k = 0.0152$. According to (2.24) and Table 2.2, Case 1 can not make the system stable, whereas the other cases can. This is verified by the simulation results shown in Figure 2.12.

The system response with a big uncertainty in the process is shown in Figure 2.13, where the process has a 130% uncertainty in its gain, i.e., $\Delta k = 0.0658$. It can be seen that Case 2 becomes unstable too, whereas Case 3 and Case 4 can still maintain system stability. This shows that in somewhat there is a real tension between good setpoint tracking and satisfactory robustness. On the other hand, Case 3, compared to Case 4, gives a better robustness though the later one has a better disturbance rejection. This is the contradiction between the robustness and the disturbance rejection as mentioned before.

### 2.8.2 Example 2: 2-DOF control

In this example, the 2-DOF control structure and single-parameter PI controller are discussed. In Figure 2.10, consider the following process

$$G(s) = \frac{1}{s}e^{-5s}$$

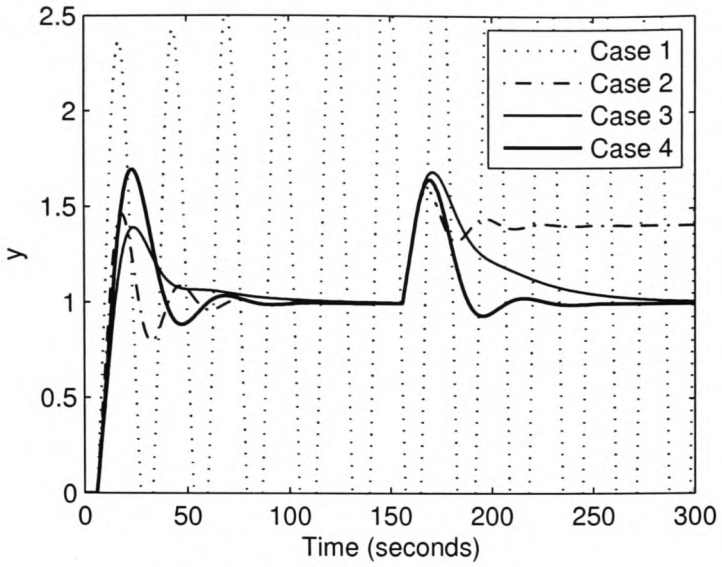



Figure 2.12: Robust responses with $\Delta k = 0.0152$.

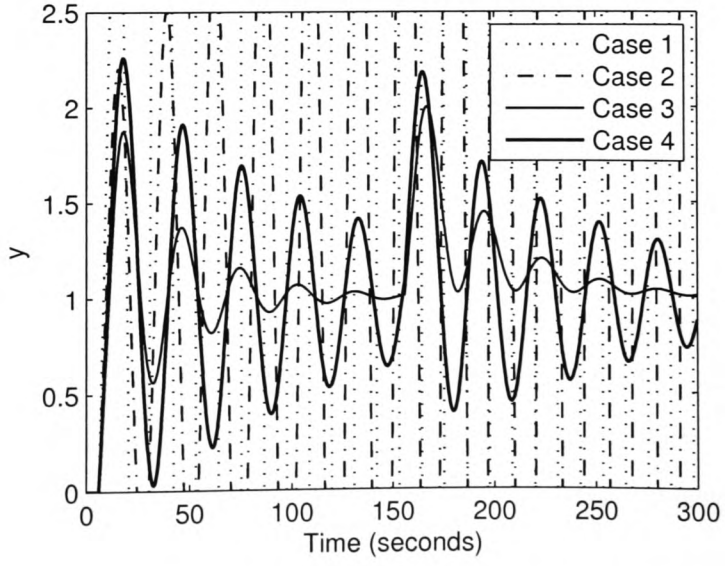



Figure 2.13: Robust responses with $\Delta k = 0.0658$.

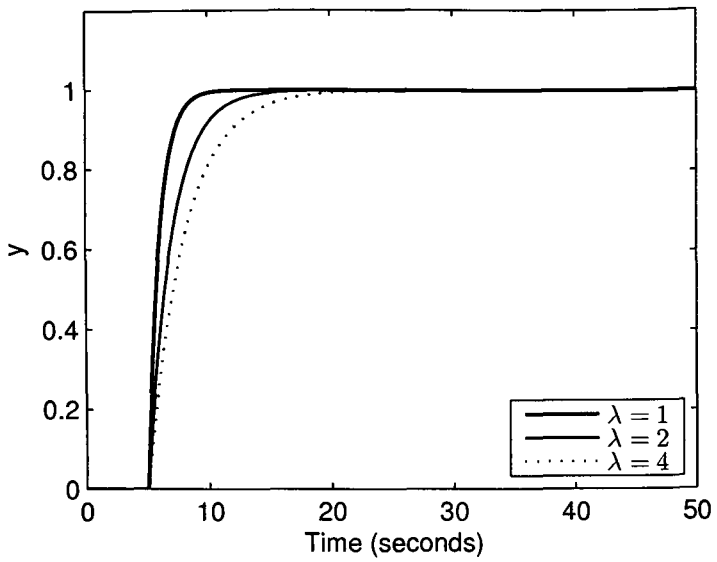



Figure 2.14: Desired setpoint responses under 2-DOF structure.

which was studied by Åström et al. (1994); Normey-Rico and Camacho (1999); Zhong and Li (2002).

When there is no disturbance and model-process mismatching in the system, the unit step setpoint responses are shown in Figure 2.14, where $\lambda = 1, 2, 4$ in $F(s)$. There is no overshoot in the responses; the smaller $\lambda$ is, the faster the response becomes (if the control energy is available). This is one of the advantages of 2-DOF control which separates the setpoint response from the disturbance response.

When the disturbance and/or model-process mismatching exist, four cases of controller $C(s)$ are compared in Table 2.3. Consider a step disturbance with magnitude $-0.1$ added into the system at time 50 seconds. Set $\lambda = 1$ in $F(s)$. The nominal responses are shown in Figure 2.15. Case 1 gives the best disturbance rejection. Case 2 gives a better disturbance response compared with Case 3 and Case 4. Simulation results for the process with uncertainty $\Delta\tau = 1.4$ in the dead time are shown in Figure 2.16. Case 2 becomes unstable, while Case 4 gives the best robustness. The robustness of Case 3 is bad, and the robustness of Case 4 is acceptable. Together with Figure 2.15, for the same kind of controller, the selection of the control parameters needs to compromise the robustness and the disturbance rejection. For different kinds of controllers, however, it is possible that one kind of

Table 2.3: Second control degree of freedom $C(s)$ for Example 2.

| | $\bar{k}_p$ | $\frac{1}{T_i}$ | $k_p$ | $T_i$ | $\bar{\alpha}$ | $\alpha$ |
|---|---|---|---|---|---|---|
| Case 1: $C^{(sub)}(s)$ | - | - | - | - | - | 4 |
| Case 2: $C(s)$ | 1.11 | 0.22 | 0.22 | 22.73 | - | - |
| Case 3: $C^{(\alpha)}(s)$ | 0.8 | 0.38 | 0.16 | 13.2 | 0.82 | 4.1 |
| Case 4: $C^{(\alpha)}(s)$ | 0.64 | 0.25 | 0.13 | 20 | 1.5 | 7.5 |

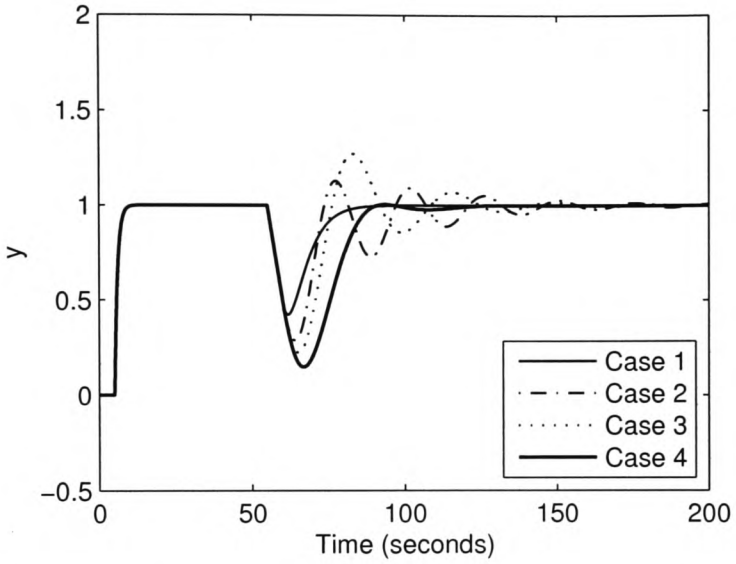



Figure 2.15: Disturbance responses with $\Delta\tau = 1.4$ under 2-DOF structure.

controller can achieve better performances on the robustness and the disturbance rejection at the same time. For example, here $C^{(sub)}(s)$ is much better than $C(s)$ or $C^{(\alpha)}(s)$. On the other hand, $C^{(sub)}(s)$ and $C^{(\alpha)}(s)$ are easy to tune the single controller parameter $\alpha$ to compromise the control performances. For $C(s)$, this is not simple. From this point of view, sometimes the single parameter PI controller, such as $C^{(\alpha)}(s)$, is more popular than the typical PI controller $C(s)$, though $C^{(\alpha)}(s)$ has more restricted control limits.

## 2.9 Summary

This chapter has discussed many issues about the control of IPDT systems by PI controllers under the classical feedback structure and 2-DOF structure. Two types of controllers, i.e., typical PI controller and single-parameter PI controller, were both

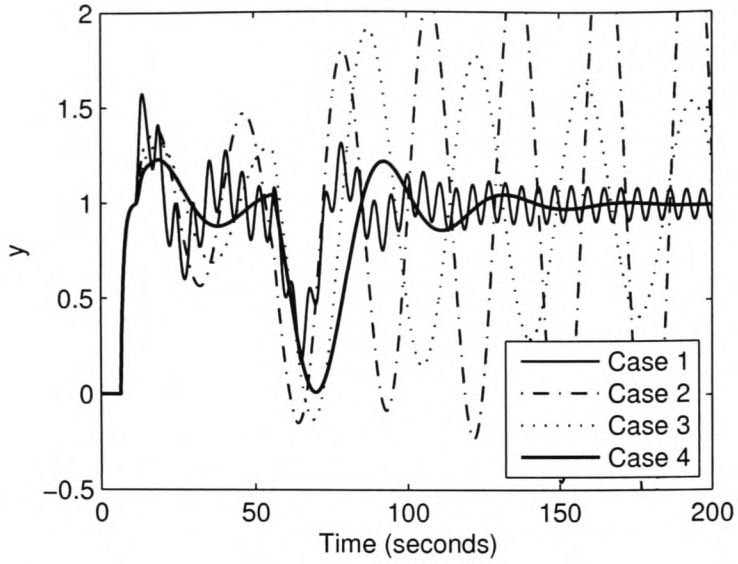



Figure 2.16: Robust responses with $\Delta\tau = 1.4$ under 2-DOF structure.

addressed.

Firstly, the stabilization region of the parameters of the PI controller was given numerically. In this region, the controller can be designed according to the specified gain and/or phase margins to achieve desired robustness. The constraint on the achievable stability margins in this system has also been obtained. Secondly, two system performance indexes, setpoint-ISE and disturbance-ISE, were also discussed quantitatively. The simulation results show the system has to compromise between the robustness, the setpoint response and the disturbance rejection.

In order to design the controller more easily, a single-parameter PI controller was deduced according to the sub-ideal disturbance rejection. All above discussed system conclusions correspond to respective modified versions which in general are more restricted. The 2-DOF structure used in this chapter decouples the setpoint response of the system from its robustness and disturbance rejection. This structure in essential is a modified Smith predictor structure.

# Chapter 3

# Implementation of Distributed Delays in Linear Control Laws

The control of time-delay systems often involves a distributed delay. This chapter provides an estimation of the minimal number of steps to implement the distributed delay into a series of discrete delays in linear control laws. This is obtained by solving an inequality with respect to the number of the implementation steps. A coarse estimation is firstly given as the initial value to solve the inequality using bisection algorithms. A minimization process as well as some other techniques are also introduced to further improve the estimation. Examples show that the proposed estimation provides a useful guide for the choice of the number of the implementation steps.

## 3.1 Introduction

Consider the dead-time system described by the following transfer function

$$G(s) = e^{-sh}P(s), \tag{3.1}$$

where $h > 0$ is the delay and $P(s)$ is a rational transfer function. Assume $P(s)$ having a minimal state-space realization

$$\begin{aligned} P(s) &= C(sI - A)^{-1}B \\ &= \left[\begin{array}{c|c} A & B \\ \hline C & 0 \end{array}\right], \end{aligned} \tag{3.2}$$

where $A \in \mathbb{R}^{n\times n}$, $B \in \mathbb{R}^{n\times 1}$ and $C \in \mathbb{R}^{1\times n}$; $(A, B)$ is stabilizable; $(C, A)$ is detectable. The control of this kind of systems often involves a distributed delay (Manitius and Olbrot, 1979; Watanabe, 1986; Wang et al., 1999; Watanabe and Ito, 1981; Palmor, 1996; Zhong, 2006; Meinsma and Zwart, 2000; Zhong, 2003b,c; Mirkin, 2003b; Zhong, 2003d) described by

$$v(t) = \int_0^h e^{A\zeta}Bu(t-\zeta)\mathrm{d}\zeta,$$

where the delay from $u \in \mathbb{R}$ to $v \in \mathbb{R}^{n\times 1}$ is distributed (integral) over a finite time $h$. The $s$-domain equivalent of the above distributed delay is the transfer function vector from $u$ to $v$ given by

$$\Pi(s) = \left(I - e^{-(sI-A)h}\right)(sI - A)^{-1}B. \tag{3.3}$$

Note that $\Pi(s)$ is the difference of two blocks $(sI-A)^{-1}B$ and $e^{-(sI-A)h}(sI-A)^{-1}B$ and all poles of $\Pi(s)$ are cancelled by its zeros.

Due to the requirement of internal stability, $\Pi(s)$ has to be approximately implemented as a stable block, instead of the difference of two unstable blocks, if $A$ has unstable eigenvalues. Conventionally $\Pi(s)$ is replaced by the sum of a series of discrete delays (Manitius and Olbrot, 1979; Watanabe and Ito, 1981; Palmor, 1996).

However, Van Assche et al. (1999) reported that this implementation method could not guarantee system stability even when a quite accurate approximation of the integral law is used, which was then studied extensively, *e.g.*, in Engelborghs et al. (2001); Santos and Mondie (2000); Van Assche et al. (1999); Richard (2003); Mirkin (2004). It was proved in Zhong (2004) that this is not the case: a distributed delay can be implemented as usual by replacing it with a series of discrete delays. There always exists a minimal number $N_{\min}$ of the implementation steps so that the system stability is guaranteed when the number of the implementation steps is larger than $N_{\min}$. Alternative methods to implement $\Pi(s)$ using rational functions were reported in Zhong (2005a,b).

Ideally what is required is a formula to calculate the minimal $N_{\min}$ so that the practical implementation could be easily done. However, $N_{\min}$ is difficult to obtain because of the complexity of the resulting system. In fact, no result is available in the literature to date. In this chapter, attempts are made to estimate $N_{\min}$. This chapter follows the framework proposed in Mirkin (2003a, 2004) to study the effect of approximation error on the system stability using the well-known small-gain theorem. A formula will be derived to find an estimation of $N_{\min}$ as small as possible to guarantee the system stability. Two cases will be considered here for the process, namely with and without process uncertainty.

The rest of the chapter is organized as follows. Preliminaries including mathematical tools and all stabilizing controllers for delay systems are reviewed in Section 3.2. The idea of using the small gain theorem is developed in Section 3.3. The main estimation results and possible improvements are given in Section 3.4. Examples are given in Section 3.5. Example 3.1 shows that the above estimation is somewhat conservative, due to the nature of the small gain theorem. Example 3.2 shows that sometimes different realizations (3.2) of the same process give very different results. Example 3.3 shows that there are cases where the exact minimal $N_{\min}$ can be found. Concluding remarks are made in Section 3.6.

## 3.2 Preliminary

### 3.2.1 Mathematical tools

Let $\mathcal{H}_2$ and $\mathcal{H}_\infty$ be the standard Hardy spaces defined on the open right-half complex plane (Zhou et al., 1996). The corresponding $\mathcal{H}_\infty$-norm of $G(s) \in \mathcal{H}_\infty$ is defined as

$$\|G(s)\|_\infty = \sup_{\omega \in \mathbb{R}} \bar{\sigma}(G(j\omega)),$$

where $\bar{\sigma}(\cdot)$ denotes the maximum singular value. The $\mathcal{H}_\infty$-norm can be induced from the $\mathcal{H}_2$-norm in $\mathcal{H}_2$ as

$$\|G(s)\|_\infty = \sup_{u \in \mathcal{H}_2} \frac{\|Gu\|_2}{\|u\|_2}.$$

For the sake of convenience, some well known facts are listed in Lemma 3.1 and will be used in the following sections.

**Lemma 3.1** *(i) Let $G(s)$ and $W(s)$ be two matrices in $\mathcal{H}_\infty$ with appropriate dimensions such that $G(s)W(s)$ is well defined, then $\|G(s)W(s)\|_\infty \leq \|G(s)\|_\infty \|W(s)\|_\infty$.*

*(ii) Let $T(s,\xi)$ be a matrix in $\mathcal{H}_\infty$ for $\xi \in [0,h]$, then $\left\|\int_0^h T(s,\xi)d\xi\right\|_\infty \leq \int_0^h \|T(s,\xi)\|_\infty d\xi$.*

*(iii) Given any real $\alpha > 0$ and a constant square matrix $A$, which can be regarded as a constant element in the Hardy space $\mathcal{H}_\infty$, then the $\mathcal{H}_\infty$-norm of $e^{\alpha A}$ satisfies*

$$||e^{\alpha A}||_\infty = ||e^{\alpha A}|| \leq e^{\alpha \mu(A)},$$

*where $||\cdot||$ and $\mu(\cdot)$ denote the 2-norm and logarithmic norm [1] of a matrix, respectively. Here $\mu(\cdot)$ takes*

$$\mu(A) = \max\left\{\mu | \mu \in \lambda\left(\frac{A^* + A}{2}\right)\right\}$$

*from Van Loan (1977); Strom (1975), where $A^*$ is the conjugate transpose of $A$ and*

[1] There are various bounds for the norm of matrix exponential. In this chapter, the bound characterized by the log norm is used.

$\lambda(\cdot)$ *denotes the eigenvalue.*

### 3.2.2 All stabilizing controllers

For the dead-time system defined in (3.1) and (3.2), if there exist $F$ and $L$ such that $A+BF$ and $A+LC$ are Hurwitz, then all stabilizing controllers for the system can be described by the predictor-observer control structure shown in Figure 3.1, where

$$Z(s) = Ce^{-Ah}\left(I - e^{-(sI-A)h}\right)(sI-A)^{-1}B \tag{3.4}$$

is a modified Smith predictor and $Q(s)$ is arbitrary but stable. Hereafter, assume $Q(s) = 0$. See Zhong (2006); Mirkin and Raskin (2003) for more details. In the nominal case, the output of the predictor is

$$\begin{aligned} y_p &= \left(P(s)e^{-hs} + Z(s)\right)u \\ &= \left(C(sI-A)^{-1}Be^{-hs} + Ce^{-Ah}\left(I - e^{-(sI-A)h}\right)(sI-A)^{-1}B\right)u \\ &= \left(C(sI-A)^{-1}Be^{-hs} + \left(Ce^{-Ah} - Ce^{-hs}\right)(sI-A)^{-1}B\right)u \\ &= Ce^{-Ah}(sI-A)^{-1}Bu, \end{aligned}$$

where the delay effect is eliminated. The observer provides the state

$$\begin{aligned} x_J &= e^{Ah}(sI-A-LC)^{-1}\left(e^{-Ah}Bu - Ly_p\right) \\ &= e^{Ah}(sI-A-LC)^{-1}\left(e^{-Ah} - LCe^{-Ah}(sI-A)^{-1}\right)Bu \\ &= e^{Ah}(sI-A-LC)^{-1}\left((sI-A)e^{-Ah}(sI-A)^{-1} - LCe^{-Ah}(sI-A)^{-1}\right)Bu \\ &= (sI-A)^{-1}Bu. \end{aligned}$$

This is the exact state of the delay-free system $P(s)$.

### 3.2.3 Implementation of $Z(s)$

As mentioned before, all poles of $Z(s)$ are cancelled by its zeros. If there exist unstable poles in system, $Z(s)$ can not be implemented directly for the internal

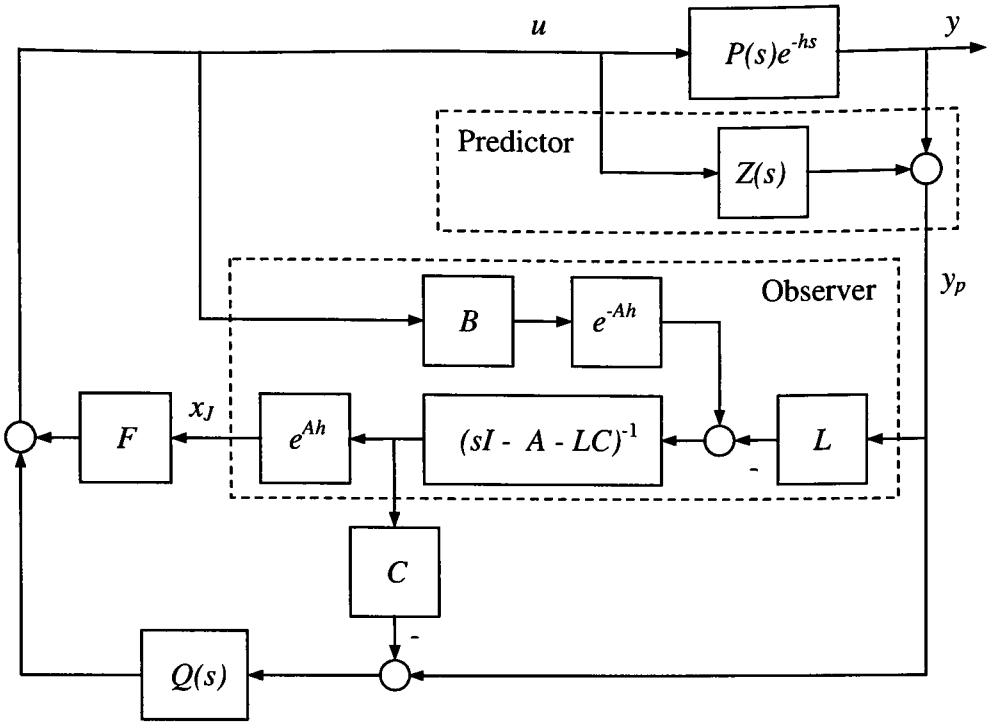



Figure 3.1: Predictor-observer structure

stability requirement. Rewrite $Z(s)$ as

$$\begin{aligned} Z(s) &= Ce^{-Ah}\left(I - e^{-(sI-A)h}\right)(sI-A)^{-1}B \\ &= Ce^{-Ah}\left(I - e^{-(sI-A)\frac{h}{N}}\right)(sI-A)^{-1}\sum_{i=0}^{N-1} e^{-i\frac{h}{N}(sI-A)}B \\ &= Ce^{-Ah}Z_1(s)B, \end{aligned}$$

where

$$Z_1(s) = \left(I - e^{-(sI-A)\frac{h}{N}}\right)(sI-A)^{-1}\sum_{i=0}^{N-1} e^{-i\frac{h}{N}(sI-A)}.$$

Integer $N$ is called the number of approximation (implementation) steps. There still exists zero-pole cancellation in $Z_1(s)$. The following implementation was proposed in Zhong (2004):

$$Z_{\epsilon 1}(s) = \frac{1-e^{-(s+\epsilon)\frac{h}{N}}}{s+\epsilon}\frac{\epsilon\frac{h}{N}}{1-e^{-\epsilon\frac{h}{N}}}\left(e^{A\frac{h}{N}}-I\right)\left(A\frac{h}{N}\right)^{-1}\sum_{i=0}^{N-1} e^{-i\frac{h}{N}(sI-A)}, \quad (3.5)$$

where $\epsilon$ is a small positive number. Clearly, $Z_{\epsilon 1}(s)$ has no zero-pole cancellation and satisfies

$$\lim_{s\to 0} Z_{\epsilon 1}(s) = \lim_{s\to 0} Z_1(s)$$

which guarantees the steady-state performance. It can be seen from (3.4) and (3.5), the distributed delay over $0 \sim h$ is converted to a series of discrete delays $0, \frac{h}{N}, \frac{2h}{N}, \cdots, \frac{N-1}{N}h$.

Note that when $A$ is singular, the term $\left(e^{A\frac{h}{N}} - I\right)\left(A\frac{h}{N}\right)^{-1}$ in $Z_{\epsilon 1}(s)$ should be replaced with $\int_0^1 e^{A\frac{h}{N}\eta} d\eta$ which can be calculated by using numerical integration methods.

## 3.3 Robust stability w.r.t. implementation error

When $Z_1(s)$ is implemented as $Z_{\epsilon 1}(s)$, denote the implementation error by

$$E_{\epsilon 1}(s) = Z_{\epsilon 1}(s) - Z_1(s).$$

That is to say, $Z_{\epsilon 1}(s)$ can be treated as an additive uncertainty $E_{\epsilon 1}(s)$ acts on $Z_1(s)$. Furthermore, $E_{\epsilon 1}(s)$ can be converted to a multiplicative uncertainty on the original system (Mirkin, 2003a). This is shown in Figure 3.2. According to the small-gain theorem (Zhou et al., 1996), the system is internally stable if

$$\|E_{\epsilon 1}(s)\|_\infty < \frac{1}{\|T_{ab}(s)\|_\infty}, \tag{3.6}$$

where $T_{ab}(s)$ is the transfer function from $b$ to $a$. Cancelling $u$ and $x_J$ in the following equations

$$\begin{aligned} x_J &= (sI - A)^{-1}Bu - e^{Ah}(sI - A - LC)^{-1}LCe^{-Ah}b \\ u &= Fx_J \\ a &= BFx_J \end{aligned}$$

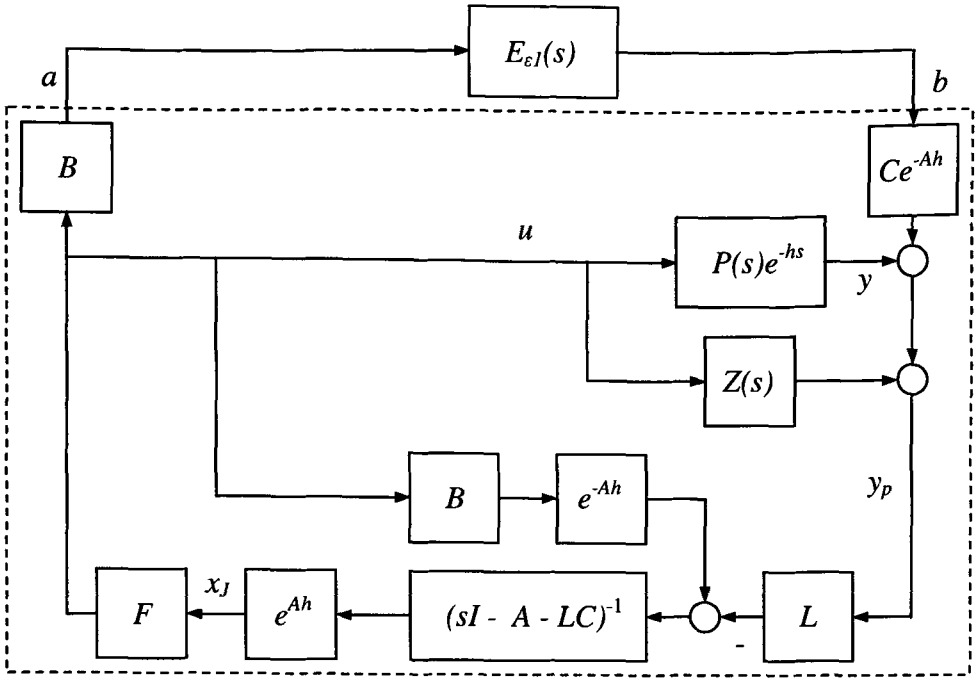



Figure 3.2: Equivalent implementation structure

yields

$$\begin{aligned} T_{ab}(s) &= -BF(sI-A-BF)^{-1}(sI-A)e^{Ah}(sI-A-LC)^{-1}LCe^{-Ah} \\ &= B\left[\begin{array}{c|c} A+BF & B \\ \hline F & I \end{array}\right] Fe^{Ah}\left[\begin{array}{c|c} A+LC & -L \\ \hline I & 0 \end{array}\right] Ce^{-Ah}. \end{aligned}$$

Rewrite the approximation error $E_{\epsilon 1}(s)$ as

$$\begin{aligned} E_{\epsilon 1}(s) &= \left(\frac{1-e^{-(s+\epsilon)\frac{h}{N}}}{s+\epsilon}\frac{\epsilon\frac{h}{N}}{1-e^{-\epsilon\frac{h}{N}}}\left(e^{A\frac{h}{N}}-I\right)\left(A\frac{h}{N}\right)^{-1}-\right. \\ &\quad \left.\left(I-e^{-(sI-A)\frac{h}{N}}\right)(sI-A)^{-1}\right)\sum_{i=0}^{N-1}e^{-i\frac{h}{N}(sI-A)} \\ &= E_{\epsilon 2}(s)Z_{wf}(s), \end{aligned}$$

with

$$E_{\epsilon 2}(s) = \frac{N}{h}\left(\frac{1-e^{-(s+\epsilon)\frac{h}{N}}}{s+\epsilon}\frac{\epsilon\frac{h}{N}}{1-e^{-\epsilon\frac{h}{N}}}\left(e^{A\frac{h}{N}}-I\right)\left(A\frac{h}{N}\right)^{-1}-\right.$$

$$\left(I - e^{-(sI-A)\frac{h}{N}}\right)(sI-A)^{-1}\Big)$$
$$= \frac{\epsilon\frac{h}{N}}{1-e^{-\epsilon\frac{h}{N}}} \frac{1-e^{-(s+\epsilon)\frac{h}{N}}}{(s+\epsilon)\frac{h}{N}} \int_0^1 e^{A\frac{h}{N}\eta} d\eta - \int_0^1 e^{-(sI-A)\frac{h}{N}\eta} d\eta, \quad (3.7)$$

and

$$Z_{wf}(s) = \frac{h}{N}\sum_{i=0}^{N-1} e^{-i\frac{h}{N}(sI-A)}. \quad (3.8)$$

It can be proved that $\lim_{N\to+\infty} \|E_{\epsilon 2}(s)\|_\infty = 0$ and $\|Z_{wf}(s)\|_\infty$ is boundary on the closed right half plane, so $\lim_{N\to+\infty} \|E_{\epsilon 1}(s)\|_\infty = 0$. Hence condition (3.6) is always met when $N \to +\infty$.

Clearly $Z_{\epsilon 1}(s)$ cannot be implemented with an infinite $N$. It is desirable to find the minimal $N_{\min}$ such that when $N > N_{\min}$ the system is stable. This will quite simplify the implementation. Here, finding $N_{\min}$ is still according to the small-gain theorem. It is known that this theorem is conservative, but arguably there is not a better way available at the moment.

If the process (3.1) has an uncertainty $\Delta_P(s) \in \mathcal{H}_\infty$, it can be considered together with $E_{\epsilon 1}(s)$, as shown in Figure 3.3. Let $E_w(s)$ be the sum of the uncertainties, *i.e.*,

$$E_w(s) = Ce^{-Ah}E_{\epsilon 1}(s)B + \Delta_P(s).$$

The system is internally stable if

$$\left\|Ce^{-Ah}E_{\epsilon 1}(s)B\right\|_\infty + \|\Delta_P(s)\|_\infty < \frac{1}{\|T_{a'b'}(s)\|_\infty}, \quad (3.9)$$

where

$$T_{a'b'}(s) = \left[\begin{array}{c|c} A+BF & B \\ \hline F & I \end{array}\right] Fe^{Ah} \left[\begin{array}{c|c} A+LC & -L \\ \hline I & 0 \end{array}\right].$$

When $\|\Delta_P(s)\|_\infty$ is known, the minimal $N_{\min}$ to guarantee (3.9) can be estimated.

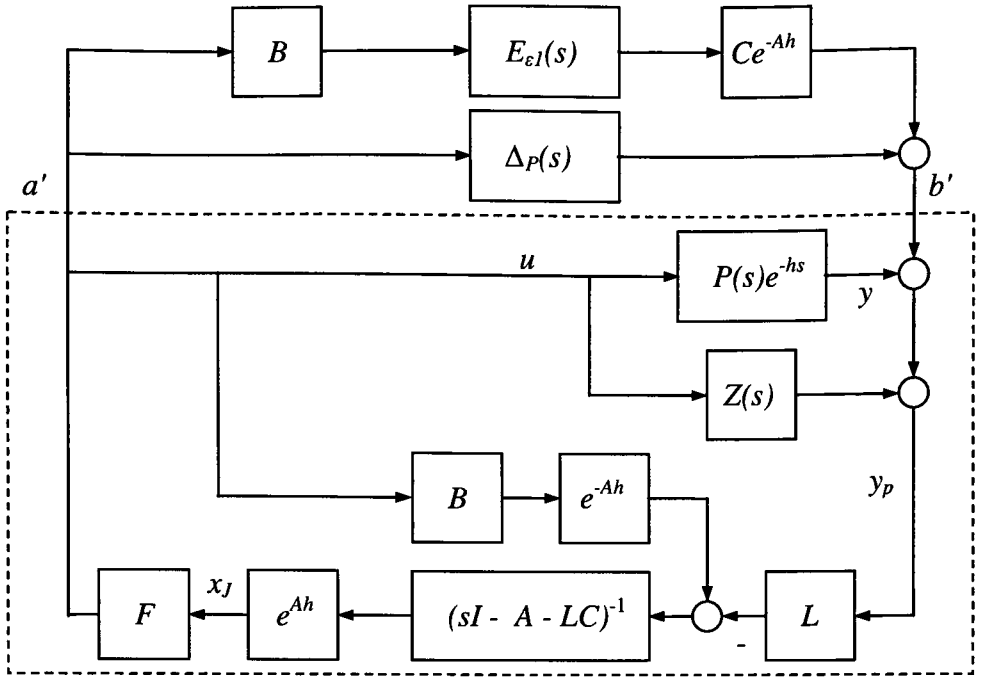



Figure 3.3: Equivalent implementation structure considering process uncertainty

## 3.4 Estimation of $N_{\min}$

This section considers the estimation of $N_{\min}$. For simplicity, the process without uncertainty is considered first. The process with uncertainty follows readily from (3.9).

### 3.4.1 Estimation of $N_{\min}$ for processes without uncertainty

For the sake of tidiness, (3.7) and (3.8) are rewritten as

$$E_{\epsilon 2}(s) = \frac{\bar{\epsilon}}{1-e^{-\bar{\epsilon}}} \frac{1-e^{-(s\frac{h}{N}+\bar{\epsilon})}}{(s\frac{h}{N}+\bar{\epsilon})} \int_0^1 e^{\bar{A}\eta} \mathrm{d}\eta - \int_0^1 e^{-(s\frac{h}{N}I-\bar{A})\eta} \mathrm{d}\eta, \tag{3.10}$$

and

$$Z_{wf}(s) = \frac{h}{N} \sum_{i=0}^{N-1} e^{-i(s\frac{h}{N}I-\bar{A})}, \tag{3.11}$$

where $\bar{A} = A\frac{h}{N}$ and $\bar{\epsilon} = \epsilon\frac{h}{N}$.

**Theorem 3.1** *The system is stable if the number N of implementation steps satis-*

*fies*

$$(\gamma_1+\gamma_2)\,\gamma_3 \;<\; \frac{1}{\|T_{ab}(s)\|_\infty}, \tag{3.12}$$

*where*

$$\gamma_1 \;=\; \frac{1.5\,\|\bar{A}\|}{\mu(\bar{A})^3}\left(\mu(\bar{A})e^{\mu(\bar{A})}-2e^{\mu(\bar{A})}+\mu(\bar{A})+2\right),$$

$$\gamma_2 \;=\; \frac{1.3\,(1-e^{-0.2\bar{\epsilon}})\left(e^{\mu(\bar{A})}-1\right)}{\mu(\bar{A})},$$

$$\gamma_3 \;=\; \frac{h}{N}\frac{e^{\mu(\bar{A})N}-1}{e^{\mu(\bar{A})}-1}.$$

**Proof**: Split (3.10) as $E_{\epsilon 2}(s) = E_{\epsilon 31}(s) + E_{\epsilon 32}(s)$, where

$$\begin{aligned}
E_{\epsilon 31}(s) &= \frac{1-e^{-s\frac{h}{N}}}{s\frac{h}{N}}\int_0^1 e^{\bar{A}\eta}\mathrm{d}\eta - \int_0^1 e^{-(s\frac{h}{N}I-\bar{A})\eta}\mathrm{d}\eta \\
&= \int_0^1 e^{\bar{A}\eta}\left(\frac{1-e^{-s\frac{h}{N}}}{s\frac{h}{N}}-e^{-s\frac{h}{N}\eta}\right)\mathrm{d}\eta \\
&= e^{\bar{A}\eta}\left(\eta\frac{1-e^{-s\frac{h}{N}}}{s\frac{h}{N}}-\frac{1-e^{-s\frac{h}{N}\eta}}{s\frac{h}{N}}\right)\Bigg|_0^1 - \\
&\quad \int_0^1\left(\eta\frac{1-e^{-s\frac{h}{N}}}{s\frac{h}{N}}-\frac{1-e^{-s\frac{h}{N}\eta}}{s\frac{h}{N}}\right)\mathrm{d}e^{\bar{A}\eta} \\
&= \int_0^1 \bar{A}e^{\bar{A}\eta}\eta\left(\frac{1-e^{-s\frac{h}{N}\eta}}{s\frac{h}{N}\eta}-\frac{1-e^{-s\frac{h}{N}}}{s\frac{h}{N}}\right)\mathrm{d}\eta,
\end{aligned} \tag{3.13}$$

and

$$\begin{aligned}
E_{\epsilon 32}(s) &= \frac{\bar{\epsilon}}{1-e^{-\bar{\epsilon}}}\frac{1-e^{-(s\frac{h}{N}+\bar{\epsilon})}}{s\frac{h}{N}+\bar{\epsilon}}\int_0^1 e^{A\frac{h}{N}\eta}\mathrm{d}\eta - \frac{1-e^{-s\frac{h}{N}}}{s\frac{h}{N}}\int_0^1 e^{\bar{A}\eta}\mathrm{d}\eta \\
&= \left(\frac{\bar{\epsilon}}{1-e^{-\bar{\epsilon}}}\frac{1-e^{-(s\frac{h}{N}+\bar{\epsilon})}}{s\frac{h}{N}+\bar{\epsilon}}-\frac{1-e^{-s\frac{h}{N}}}{s\frac{h}{N}}\right)\int_0^1 e^{\bar{A}\eta}\mathrm{d}\eta.
\end{aligned} \tag{3.14}$$

The small positive number $\epsilon$ does not affect $E_{\epsilon 31}$ but only affects $E_{\epsilon 32}$. The $\mathcal{H}_\infty$-norms of (3.11), (3.13) and (3.14) satisfy the following inequalities respectively,

$$\begin{aligned} \|Z_{wf}(s)\|_\infty &\leq \frac{h}{N}\sum_{i=0}^{N-1}\left\|e^{-i(s\frac{h}{N}I-\bar{A})}\right\|_\infty \\ &= \frac{h}{N}\sum_{i=0}^{N-1}\left\|e^{i\bar{A}}\right\|, \end{aligned}$$

$$\begin{aligned} \|E_{\epsilon 31}(s)\|_\infty &\leq \int_0^1 \left\|\bar{A}e^{\bar{A}\eta\eta}\left(\frac{1-e^{-s\frac{h}{N}\eta}}{s\frac{h}{N}\eta}-\frac{1-e^{-s\frac{h}{N}}}{s\frac{h}{N}}\right)\right\|_\infty \mathrm{d}\eta \\ &\leq \int_0^1 \|\bar{A}\|\left\|e^{\bar{A}\eta}\right\|\eta\left\|\frac{1-e^{-s\frac{h}{N}\eta}}{s\frac{h}{N}\eta}-\frac{1-e^{-s\frac{h}{N}}}{s\frac{h}{N}}\right\|_\infty \mathrm{d}\eta, \end{aligned}$$

$$\begin{aligned} \|E_{\epsilon 32}(s)\|_\infty &\leq \left\|\frac{\bar{\epsilon}}{1-e^{-\bar{\epsilon}}}\frac{1-e^{-(s\frac{h}{N}+\bar{\epsilon})}}{s\frac{h}{N}+\bar{\epsilon}}-\frac{1-e^{-s\frac{h}{N}}}{s\frac{h}{N}}\right\|_\infty \left\|\int_0^1 e^{\bar{A}\eta}\mathrm{d}\eta\right\| \\ &\leq \left\|\frac{\bar{\epsilon}}{1-e^{-\bar{\epsilon}}}\frac{1-e^{-(s\frac{h}{N}+\bar{\epsilon})}}{\left(s\frac{h}{N}+\bar{\epsilon}\right)}-\frac{1-e^{-s\frac{h}{N}}}{s\frac{h}{N}}\right\|_\infty \int_0^1 \left\|e^{\bar{A}\eta}\right\|\mathrm{d}\eta. \end{aligned}$$

In the above two inequalities, denote

$$\rho_1(s) = \frac{1-e^{-s\frac{h}{N}\eta}}{s\frac{h}{N}\eta}-\frac{1-e^{-s\frac{h}{N}}}{s\frac{h}{N}},$$

$$\rho_2(s) = \frac{\bar{\epsilon}}{1-e^{-\bar{\epsilon}}}\frac{1-e^{-(s\frac{h}{N}+\bar{\epsilon})}}{\left(s\frac{h}{N}+\bar{\epsilon}\right)}-\frac{1-e^{-s\frac{h}{N}}}{s\frac{h}{N}}.$$

$\rho_1(s)$ is the difference of two zero-order holder filters, and its norm, which is a function of $\eta$, is shown as the dashed line in Figure 3.4. A straight line $1.5(1-\eta)$ is used to approximate this curve from the upper side. See the solid line in Figure 3.4. Similarly, the norm of $\rho_2(s)$ and its approximation $1.3\,(1-e^{-0.2\bar{\epsilon}})$ are shown in Figure 3.5. Note that $\bar{\epsilon}$ is usually very small, so this approximation is quite accurate. The approximation of the norm of matrix exponential is a very complicated problem

and there are many approaches available (Van Loan, 1977). Here the logarithmic norm as given in Lemma 3.1 is adopted. As a result, $\|Z_{wf}(s)\|_\infty$, $\|E_{\epsilon 31}(s)\|_\infty$ and $\|E_{\epsilon 32}(s)\|_\infty$ can be further approximated as

$$\begin{aligned}\|Z_{wf}(s)\|_\infty &\leq \frac{h}{N}\sum_{i=0}^{N-1} e^{\mu(\bar{A})i} \\ &= \frac{h}{N}\frac{e^{\mu(\bar{A})N}-1}{e^{\mu(\bar{A})}-1},\end{aligned}$$

$$\begin{aligned}\|E_{\epsilon 31}(s)\|_\infty &\leq \int_0^1 \|\bar{A}\| e^{\mu(\bar{A})\eta}\eta(1-\eta)1.5\mathrm{d}\eta \\ &= \frac{1.5\|\bar{A}\|}{\mu(\bar{A})^3}\left(\mu(\bar{A})e^{\mu(\bar{A})} - 2e^{\mu(\bar{A})} + \mu(\bar{A}) + 2\right),\end{aligned} \tag{3.15}$$

and

$$\begin{aligned}\|E_{\epsilon 32}(s)\|_\infty &\leq \int_0^1 e^{\mu(\bar{A})\eta}\mathrm{d}\eta 1.3\left(1-e^{-0.2\bar{\epsilon}}\right) \\ &= \frac{1.3\left(1-e^{-0.2\bar{\epsilon}}\right)\left(e^{\mu(\bar{A})}-1\right)}{\mu(\bar{A})},\end{aligned} \tag{3.16}$$

which are denoted in order as $\gamma_3$, $\gamma_1$, and $\gamma_2$.

Finally, since

$$\begin{aligned}\|E_{\epsilon 1}(s)\|_\infty &= \|E_{\epsilon 2}(s)Z_{wf}(s)\|_\infty \\ &= \|(E_{\epsilon 31}(s) + E_{\epsilon 32}(s))Z_{wf}(s)\|_\infty \\ &\leq (\|E_{\epsilon 31}(s)\|_\infty + \|E_{\epsilon 32}(s)\|_\infty)\|Z_{wf}(s)\|_\infty \\ &\leq (\gamma_1+\gamma_2)\gamma_3,\end{aligned} \tag{3.17}$$

according to (3.6), the system is stable if (3.12) is satisfied. This completes the proof. □

**Remark 3.1** *Note that $E_{\epsilon 31}(s)$ is not related to $\epsilon$, and $E_{\epsilon 32}(s)$ disappears when $\epsilon \to 0$. A small $\epsilon$ means a small $\gamma_2$. Usually $\epsilon$ is chosen to be very small, thus $\gamma_2$ is*

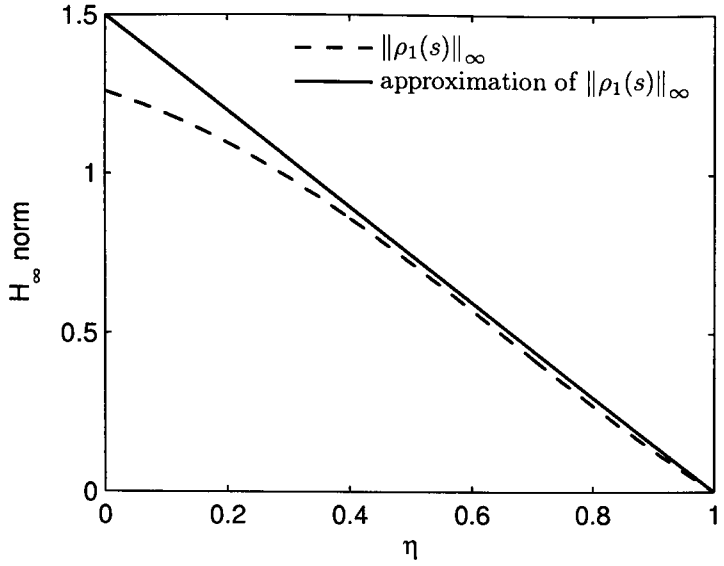



Figure 3.4: $\mathcal{H}_\infty$-norm of $\rho_1(s)$ and its approximation.

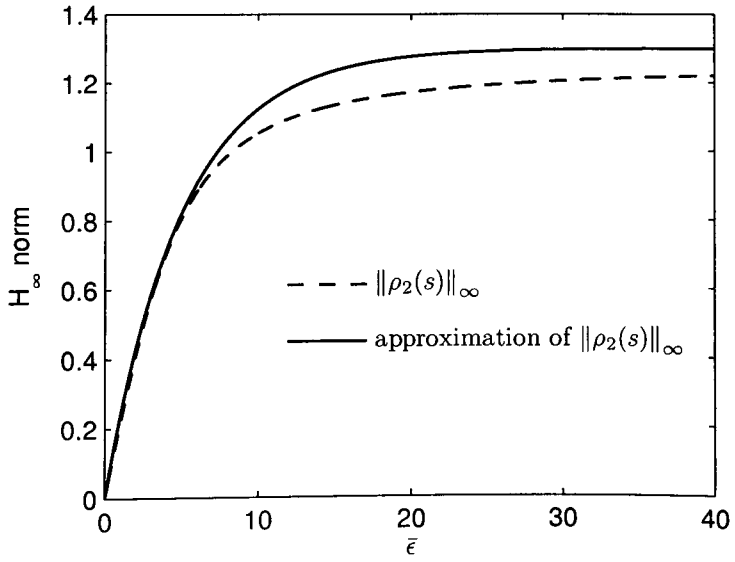



Figure 3.5: $\mathcal{H}_\infty$-norm of $\rho_2(s)$ and its approximation.

*often negligible compared to* $\gamma_1$.

Equation (3.12) is highly nonlinear and difficult to solve. The following corollary gives a more conservative estimation, which enables the use of the bisection algorithm to be proposed in the next subsection.

**Corollary 3.1** *The system is stable if the implementation steps* $N > N_0$ *where*

$$N_0 = \left\lceil h\left(0.375\left\|A\right\| + 0.26\epsilon\right)\left\|T_{ab}(s)\right\|_\infty \frac{e^{\mu(A)h} - 1}{\mu(A)} \right\rceil, \tag{3.18}$$

*where* $\lceil\cdot\rceil$ *is the ceiling function.*

**Proof**: In (3.15), $\eta(1-\eta)$ has the maximum value of 0.25. In (3.16), $1-e^{-0.2\bar{\epsilon}} \leq 0.2\bar{\epsilon}$. Hence,

$$\begin{aligned} \gamma_1 &\leq \frac{0.375\left\|\bar{A}\right\|\left(e^{\mu(\bar{A})} - 1\right)}{\mu(\bar{A})} \\ &= \gamma_1', \end{aligned}$$

$$\begin{aligned} \gamma_2 &\leq \frac{0.26\bar{\epsilon}\left(e^{\mu(\bar{A})} - 1\right)}{\mu(\bar{A})} \\ &= \gamma_2'. \end{aligned}$$

The system is stable if

$$\left(\gamma_1' + \gamma_2'\right)\gamma_3 < \frac{1}{\left\|T_{ab}(s)\right\|_\infty}.$$

Solving the above inequality gives the results in Corollary 3.1. This completes the proof. □

Because $\gamma_1 \leq \gamma_1'$ and $\gamma_2 \leq \gamma_2'$, in general $N_0$ is larger than the minimal solution of $N$ in (3.12). $N_0$ can be regarded as an upper bound of $N_{\min}$.

### 3.4.2 Estimation for processes with uncertainty

Theorem 3.1 gives the estimation of the minimal $N_{\min}$ for the processes without uncertainty. As for the processes with uncertainty, Theorem 3.2 follows directly

from Theorem 3.1, the inequality (3.9) and the relation

$$\begin{aligned} \|E_\epsilon(s)\|_\infty &= \left\|Ce^{-Ah}E_{\epsilon 1}(s)B\right\|_\infty \\ &\leq \left\|Ce^{-Ah}\right\| \|B\| \|E_{\epsilon 1}(s)\|_\infty . \end{aligned}$$

**Theorem 3.2** *For a process with uncertainty $\Delta_P(s)$, suppose*

$$1 - \|T_{a'b'}(s)\|_\infty \|\Delta_P(s)\|_\infty > 0.$$

*The system is stable if $N$ satisfies*

$$\left\|Ce^{-Ah}\right\| \|B\| (\gamma_1 + \gamma_2)\gamma_3 < \frac{1}{\|T_{a'b'}(s)\|_\infty} - \|\Delta_P(s)\|_\infty .$$

### 3.4.3 Bisection algorithm

It can be proved that $(\gamma_1 + \gamma_2)\gamma_3$ is monotonically decreasing with respect to $N$. This makes finding an estimation of the minimal $N_{\min}$, $\hat{N}_{\min}$, satisfying inequality (3.12) much easier. The following bisection searching algorithm can be used to search for the solution. The coarse $N_0$ from (3.18) is used as the initial searching value.

**Algorithm 1** Bisection searching algorithm

Initial value $N_0$;

1. If $N = 1$ satisfies (3.12), then stop and output $\hat{N}_{\min} = 1$;
2. Let $N_1 = 1$, $N_2 = N_0$;
3. If $N_2 - N_1 \leq 1$, stop and output $\hat{N}_{\min} = N_2$;
4. Substitute $N = \lceil \frac{N_1+N_2}{2} \rceil$ into (3.12). If it holds then let $N_2 = N$, otherwise let $N_1 = N$. Go to Step 3.

Algorithm 1 needs only elementary function evaluations and converges much faster than the conventional Newton algorithms for nonlinear equations since it searches the solution only on a finite number of integers. Its effectiveness and efficiency will be illustrated by examples in the next section.

### 3.4.4 Minimization of $\|T_{ab}(s)\|_\infty$

From (3.12), a smaller $\|T_{ab}(s)\|_\infty$ implies that a bigger $(\gamma_1+\gamma_2)\gamma_3$, *i.e.*, a bigger implementation error, is allowable. Therefore a suitable choice of $F$ and $L$ which minimizes $\|T_{ab}\|_\infty$ will result in give a smaller $N_{\min}$ in general. Solving the following minimization problem is reasonable as a pre-process for the estimation of the minimal $N_{\min}$.

**Problem 3.1**

$$\begin{aligned} & \min_{F,L} \|T_{ab}(s)\|_\infty \\ = & \min_{F,L} \left\| B \left[\begin{array}{c|c} A+BF & B \\ \hline F & I \end{array}\right] Fe^{Ah} \left[\begin{array}{c|c} A+LC & -L \\ \hline I & 0 \end{array}\right] Ce^{-Ah} \right\|_\infty \\ s.t. \quad & A+BF \text{ and } A+LC \text{ are Hurwitz.} \end{aligned}$$

The above problem is a nonlinear global minimization which is usually difficult to solve. However a local minimum is always easy to find.

## 3.5 Examples

**Example 3.1** Consider a simple process with $A=1$, $B=1$, $C=1$, $h=1$, i.e., $P(s)=e^{-s}\frac{1}{s-1}$ is unstable.

Design $\epsilon=0.1$, $L=-2$ and $F=-2$. Then $\|T_{ab}(s)\|_\infty = \left\|\frac{-4(s-1)}{(s+1)^2}\right\|_\infty = 4$. From (3.18), $N_0=3$. The use of bisection algorithm gives $\hat{N}_{\min}=2$. Actually, simulations show that the system is stable when $N=1$.

**Example 3.2** Consider the system

$$A=\begin{bmatrix} -1 & 0.5 \\ 4 & 0 \end{bmatrix}, \quad B=\begin{bmatrix} 0.5 \\ 0 \end{bmatrix}, \quad C=\begin{bmatrix} 0 & 0.5 \end{bmatrix}, \quad h=1,$$

that is, $P(s)=e^{-s}\frac{1}{(s-1)(s+2)}$.

Take $\epsilon = 0.01$. Let $F = \begin{bmatrix} f_1 & f_2 \end{bmatrix}$ and $L = \begin{bmatrix} l_1 & l_2 \end{bmatrix}^T$, then

$$\det(sI - (A + BF)) = s^2 - (0.5f_1 - 1)s - 4(0.5 + 0.5f_2),$$
$$\det(sI - (A + LC)) = s^2 + (1 - 0.5l_2)s - 0.5l_2 - 4(0.5 + 0.5l_1).$$

The requirements that $A + BF$ and $A + LC$ are Hurwitz can be written as

$$f_1 < 2, \quad f_2 < -1, \quad l_2 < 2, \quad l_1 < -\frac{l_2}{4} - 1,$$

and Problem 3.1 can be formulated as

$$\min_{s.t. \quad f_1<2,\ f_2<-1,\ l_2<2,\ l_1<-\frac{l_2}{4}-1} \|T_{ab}(s)\|_\infty$$

The MATLAB function 'fmincon' gives a minimal value $\|T_{ab}(s)\|_\infty = 46.48$ when

$$F = \begin{bmatrix} -16.54 & -4.93 \end{bmatrix}, \quad L = \begin{bmatrix} -0.85 \\ -17.39 \end{bmatrix}$$

for given initial values $\begin{bmatrix} 0 & -2 \end{bmatrix}$ and $\begin{bmatrix} -2 \\ 0 \end{bmatrix}$ respectively. As a result, $N_0 = 203$ is obtained from (3.18). The bisection algorithm solves (3.12) to give $\hat{N}_{\min} = 136$. However, the simulation shows that the system is stable even when $N = 2$.

There are two reasons for this discrepancy. One is that the small-gain theorem is conservative as mentioned before; the other is that $\|E_{\epsilon 1}(s)\|_\infty$ is enlarged too much in (3.17), which can be illustrated by the following two facts:

1. If $\left\|e^{i\bar{A}}\right\|$ is not approximated as $e^{\mu(A)i}$ in $\|Z_{wf}(s)\|_\infty$ in Theorem 3.1, but is computed directly, then $\hat{N}_{\min} = 113$. The drawback is that the computation involved is too demanding because of the need to compute the norm of matrix exponential many times.

2. If $P(s)$ is realized as

$$A = \begin{bmatrix} 1 & 0 \\ 0 & -2 \end{bmatrix}, \quad B = \begin{bmatrix} 1 \\ 1 \end{bmatrix}, \quad C = \begin{bmatrix} \frac{1}{3} & -\frac{1}{3} \end{bmatrix},$$

then the corresponding minimal $\|T_{ab}(s)\|_\infty = 57$. This results in $N_0 = 74$ and $\hat{N}_{\min} = 50$. In this case, $A$ is symmetric, which satisfies $\left\|e^{\alpha A}\right\| = e^{\alpha\mu(A)}$, so $\|E_{\epsilon 1}(s)\|_\infty$ is less enlarged. Hence, whenever possible, a state-space realization with a symmetric $A$ should be adopted.

**Example 3.3** Consider the following system

$$A = \begin{bmatrix} 0 & 1 \\ 0 & -1.3 \end{bmatrix}, \quad B = \begin{bmatrix} 0 \\ 1 \end{bmatrix}, \quad C = \begin{bmatrix} 1 & 0 \end{bmatrix}, \quad h = 0.5,$$

that is, $P(s) = e^{-0.5s}\frac{1}{s(s+1.3)}$.

Let $\epsilon = 0.01$ and take

$$F = \begin{bmatrix} -1.3 & 0 \end{bmatrix}, \quad L = \begin{bmatrix} -1.3 \\ 0 \end{bmatrix}$$

as the initial points for minimizing $\|T_{ab}(s)\|_\infty$. It follows that $\|T_{ab}(s)\|_\infty = 1.52$, $N_0 = 1$. Note that $N \geq 1$, thus $N_0 = 1$ is the least approximation step. This example shows that the estimation in Corollary 3.1 is accurate in this example. However if one chooses

$$F = \begin{bmatrix} -30 & 0 \end{bmatrix}, \quad L = \begin{bmatrix} -30 \\ 0 \end{bmatrix}$$

as the initial points, then $\|T_{ab}(s)\|_\infty = 150.35$, $N_0 = 25$, $\hat{N}_{\min} = 17$. This shows that the choice of of $F$ and $L$ is important to the estimation in Theorem 3.1.

## 3.6 Summary

For the implementation of distributed delays using discrete delays, this chapter has given an inequality as well as an upper bound for the minimal number of implementation steps which retains system stability. The bisection technique has been applied to solve the integer solution of the inequality. A pre-process of the minimization problem and other techniques have also been given as a remedy to the conservativeness of the estimation result. It has also been pointed out that the realization with a symmetric $A$-matrix is helpful to reduce the number of implementation steps.

The main results obtained are based on the small gain theorem which provides a sufficient but not necessary condition for system stability. This results in the obtained the estimation for the minimal approximation step $N_{\min}$ is somewhat conservative, but is useful for implementation. If a new stability criterion for time-delay systems emerges, then it is possible to get a better estimation of the minimal number of implementation steps. This still remains a challenge to be solved.

# Chapter 4

# Assignment of Network-induced Delay and its Application in Networked Predictive Control Systems

This chapter proposes an improved network-induced delay compensation strategy for networked predictive control systems. In this strategy, the network-induced delay can be partially assigned according to system stability requirements. The proposed method conveniently deals with the data-packet dropout and disorder on communication networks. Such delay assignment is necessary when considering system stability which places some limits on the network-induced delay. Several stability criteria are given for the closed-loop system with random network-induced delay, and a resulting implementation algorithm is also provided. A numerical example demonstrates the effectiveness of the proposed method.

## 4.1 Introduction

Much attention has been paid recently to the study of networked control systems (NCSs) in which the actuator and the sensor are connected to the controller via a communication network (Zhang et al., 2001; Tipsuwan and Chow, 2003; Yang, 2006). With the network, it is convenient to realize distributed control, remote control, etc. The network, however, also brings some obvious problems to the system. For example, network-transmission delay, data-packet disorder or data-packet dropout generally degrades the system performance and even makes the system unstable.

The network-transmission delay has been be modeled as constant, Markov chains, stochastic or completely random according to different network characteristics or scheduling strategies (Zhang et al., 2001; Montestruque and Antsaklis, 2004; Hespanha, 2006). Then NCSs are often treated as time-varying delay systems, and the main task is to analyze its delay-dependent stability or to design a controller such that the system can tolerate as large a network-induced delay range as possible (Yang et al., 2003; Yue and Han, 2005; He et al., 2007). In these control methods, the system just passively suffers the impact of the network-induced delay rather than actively compensates for it.

An effective way to tackle the network-induced delay is to use prediction. Exploiting the property that the network transmits data in the form of a data-packet so that a sequence of historical data or future predictions of the system signals can be transmitted at the same time, a networked predictive control (NPC) method has been developed very recently (Liu et al., 2005; Kim et al., 2006a; Liu et al., 2006, 2007b), where the prediction is based on either the polynomial model or the state-space model of the process. The NPC method mainly includes two steps: i) a sequence of future control predictions is generated at the controller node and then transmitted to the actuator node; ii) according to the measured network-induced delay, the actuator chooses the appropriate element from the received control prediction sequence as the actual control input. Thus, the impact of the network-induced delay is compensated. The resulting closed-loop system is translated to a switched system, but due to the randomness of the network, the system stability has to be subject to arbitrary switching of all subsystems (a common Lyapunov function

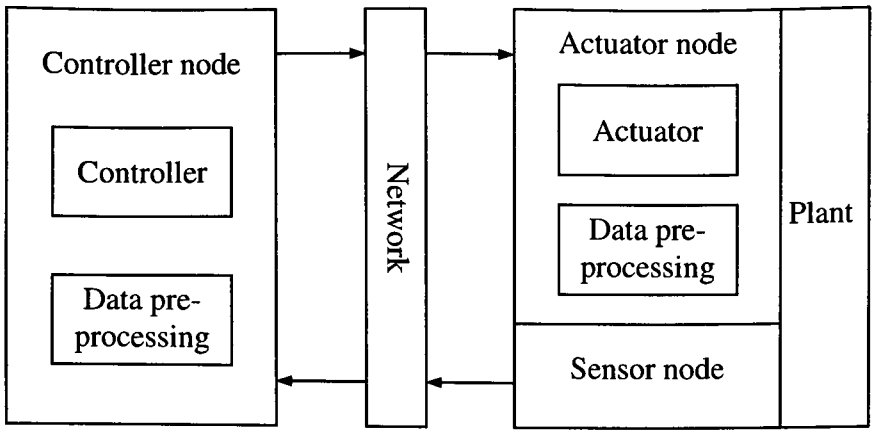



Figure 4.1: General NCS framework.

existing for them).

This chapter studies the above delay-compensation strategy and modifies it so that the closed-loop switched system has a designable switching signal. It is known that under some switching laws a switching system may be stable even if all subsystems do not possess a common Lyapunov function (Sun and Ge, 2004). Average dwell time method is an effective tool to design such switching laws (Hespanha and Morse, 1999; Zhai et al., 2001; Lin et al., 2003; Sun et al., 2006). Its main idea is to limit the switching frequency by increasing the average dwell time on individual subsystems. This chapter investigates the application of the average dwell time method in the improved NPCS to relax the restrictions that the original delay-compensation strategy imposes on the subsystems. This chapter focuses on the improvement of the delay-compensation strategy in NPCSs and its implementation considering the average dwell time, rather than on the issue of the design of controller.

## 4.2 General framework of NCSs

A general framework of the networked control system is shown in Figure 4.1, where the control loop is closed through two network channels, the controller-to-actuator channel or forward channel and the sensor-to-controller channel or feedback channel.

In this chapter, the system is discussed in discrete-time system context. Any action in the sensor node, in the controller node or in the actuator node takes place at

the sampling instant $kh$ where $k = 0, 1, 2, \cdots$ is integer and $h$ is the sampling period. That is to say, these three nodes are all time-driven. For the sake of tidiness, $h$ in the notation $kh$ is often omitted without causing confusion. The data transmitted in the network is in the form of network packet, and hence it is also called packet or data-packet in context. This packet includes not only the fundamental system signal, for example the plant output $y(k)$, the plant state $x(k)$ or the control input $u(k)$, but also includes its corresponding time-stamp, the sampling instant at which the data is sent to the network. Again for the sake of tidiness, the time-stamp is not explicitly expressed in the notation of data.

## 4.3 Data transmission on networks

In NCSs, information exchange between the control node and the actuator node or the sensor node is subject to various characteristics of the network. From the control perspective, two issues of the network, i.e., the network-transmission delay and the data dropout, rather than the network architecture, protocol and scheduling, are of more interest.

### 4.3.1 Network-transmission delay

The network-transmission delay denotes the time that a data-packet takes to transmit in the network. This time is usually random due to the uncertainties of the network circumstances, such as the network load, priorities of the other ongoing communications and electrical disturbances (RAY, 1987). For the same reason, two sources of network-transmission delay, namely $\sigma_{ca}(k)$ in the controller-to-actuator channel and $\sigma_{sc}(k)$ in the sensor-to-controller channel, usually possess different characteristics. Any computation time of dealing with data sending and receiving with the network can be integrated into either $\sigma_{ca}(k)$ or $\sigma_{sc}(k)$ without loss of generality (Zhang et al., 2001). Correspondingly, the round-trip network transmission delay $\sigma(k)$ denotes the time duration in two network channels in one control cycle (sensor-controller-actuator). Note that in this chapter the delay represents the integer multiplication of the real time-delay w.r.t. the system sampling period $h$. For a

general communication network, it is reasonable to make the following assumption, or otherwise the system becomes open-loop.

**Assumption 4.1** *$\sigma_{ca}(k)$ and $\sigma_{sc}(k)$ are random but with upper bound $\bar{\sigma}_{ca}$ and $\bar{\sigma}_{sc}$ respectively.*

Due to its randomness, the value of the network-transmission delay of a data-packet is unknown before the packet is received at the controller node or at the actuator node. The accurate measurement of this delay also depends on the clock synchronousness between the two sides of the network. As shown in Figure 4.2 where the sensor node clock (also the actuator node clock) is temporarily denoted as $k^{(s)}$ and the controller node clock as $k^{(c)}$, $y(k_1^{(s)})$ is sent out from the sensor node at the sampling instant $k_1^{(s)}$ and is received at the controller node at the sampling instant $k_2^{(c)}$, and $u(k_2^{(c)})$ is sent out from the controller node at $k_2^{(c)}$ and is received at the actuator node at $k_3^{(s)}$. The network-transmission delay of $y(k_1^{(s)})$ and $u(k_2^{(c)})$ is calculated as

$$\sigma_{sc}(k_2^{(c)}) = k_2^{(c)} - k_1^{(c)} \tag{4.1}$$

$$\sigma_{ca}(k_3^{(s)}) = k_3^{(s)} - k_2^{(s)}, \tag{4.2}$$

where $k_1^{(c)}$ and $k_2^{(s)}$ are the equivalent sampling instants of $k_1^{(s)}$ and $k_2^{(c)}$ w.r.t. the controller node clock and the sensor node clock, respectively. Bear in mind that the packet is transmitted along with its time-stamp. Here the two time-stamps are $k_1^{(s)}$ and $k_2^{(c)}$. If the two clocks are identical, i.e., $k^{(s)} = k^{(c)}$, (4.1) and (4.2) become to

$$\sigma_{sc}(k_2^{(c)}) = k_2^{(c)} - k_1^{(s)}$$

$$\sigma_{ca}(k_3^{(s)}) = k_3^{(s)} - k_2^{(c)}.$$

In practice, it is almost unavoidable that the clocks $k^{(s)}$ and $k^{(c)}$ are asynchronous, so they need to be synchronized. Although clock synchronization is complicated and is a research area in itself, some effective methods are still available (Nilsson, 1998; Johannessen, 2004, and the references therein). The main idea is to estimate the clock offset (skew) by sending clock-read request forth and clock-read result back

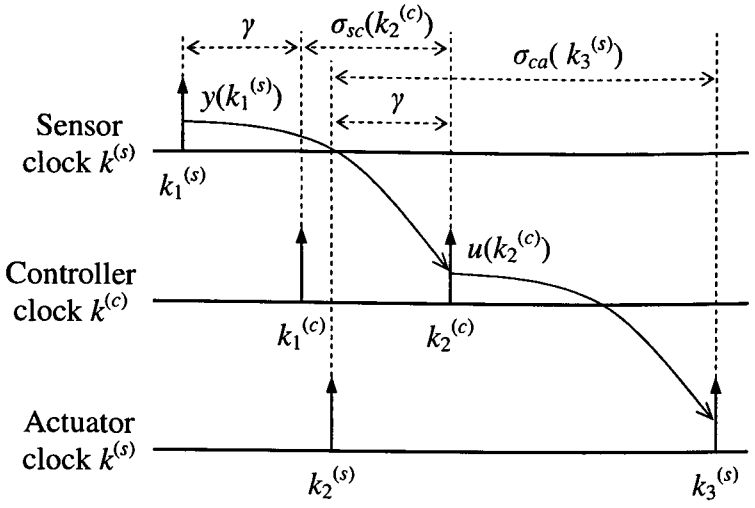



Figure 4.2: Network-transmission delay under asynchronous clocks.

between two nodes which want to synchronize their clocks. Denote the offset of $k^{(s)}$ and $k^{(c)}$ be

$$\gamma = k^{(c)} - k^{(s)}.$$

After synchronization, $\gamma$ is determined and (4.1) and (4.2) are correspondingly adjusted as

$$\sigma_{sc} = k_2^{(c)} - k_1^{(c)} = k_2^{(c)} - k_1^{(s)} - \gamma$$

$$\sigma_{ca} = k_3^{(s)} - k_2^{(s)} = k_3^{(s)} - k_2^{(c)} + \gamma.$$

Note that re-synchronization of clocks might be needed after a while because of the clock drift.

For system analysis purposes, the round-trip network-transmission delay $\sigma(k)$, rather than the unilateral delay $\sigma_{ca}(k)$ or $\sigma_{sc}(k)$, is of more concern. Yang (2006) pointed out that it is the round-trip network-transmission delay and data-packet dropout, rather than the clock offset, which causes deterioration in system performance. It can be seen from Figure 4.2 that in the control cycle $k_1^{(s)}$ to $k_3^{(s)}$ the round-trip network-transmission delay $\sigma(k_3^{(s)})$, the sum of $\sigma_{sc}(k_2^{(c)})$ and $\sigma_{ca}(k_3^{(s)})$, can be obtained directly from

$$\sigma(k_3^{(s)}) = k_3^{(s)} - k_1^{(s)}$$

which is independent of the controller node clock. This means that it does not matter whether or not there exists clock offset between the sensor clock and the controller clock, so clock synchronization is not necessary. Note that in this case, the time-stamp in packet $u(k_2^{(c)})$ should keep using $k_1^{(s)}$ rather than $k_2^{(c)}$. However, in systems where the design of the controller depends on $\sigma_{ca}(k)$ or $\sigma_{sc}(k)$, for example the predictive control, clock synchronization is still indispensable. One exception of this can be found in Hu et al. (2007).

### 4.3.2 Data dropout

The phenomenon of data being dropped out when it is transmitted in the network is another significant feature of NCSs compared to the traditional control systems. Data dropout takes place when the transmission fails in the physical network links or the network buffer overflows due to congestion. Although some network protocols, such as TCP, require re-delivering the data when it is dropped out, this has little benefit because the re-delivered data might be too old for control purposes, but contrarily aggravates the network burden. For this reason, if a data-packet is eventually received over a long time, for example greater than the upper bound in Assumption 4.1, it is discarded intentionally. This essentially amounts to a data dropout.

Data dropout is often described using the dropout rate, which can be determined by counting the received data-packets from one side of the network out of the total packets sent out from the other side of the network during a period of time. As a result, the system analysis is usually subject to probability theory (Montestruque and Antsaklis, 2004). In this chapter, the number of consecutive dropouts is instead adopted to describe the data dropout. For the same reason as that in Assumption 4.1, it is also reasonable to make the following assumption.

**Assumption 4.2** *The number of consecutive data-packet dropouts in the controller-to-actuator channel or the sensor-to-actuator channel is not greater than* $\varsigma_{ca}$ *or* $\varsigma_{sc}$ *respectively.*

## 4.4 Data pre-processing

A data-packet dropout in the network results in a failure of the controller node or the actuator node to receive a data-packet at a future sampling instant. On the other hand, due to the randomness of the network-transmission delay, it is possible that data-packets which are sent at different sampling instants might be received at the same sampling instant. In other words, this implies that there is no packet received at some other sampling instants.

From the above analysis, at each sampling instant, data receiving encounters one of the following three situations:

- One data-packet received;
- Multi data-packets received; or
- No data-packet received.

Therefore, in NCSs, compared to the traditional control systems, an extra data pre-processing unit is needed at the controller node and at the actuator node, as shown in Figure 4.1. This unit is responsible for providing a data-packet for the controller or the actuator, especially when a failure of data receiving happens. In this chapter, data pre-processing works as follows. Firstly, a buffer is used to store all received data-packets during a period of time. The data to be used in the controller or actuator is then chosen from this buffer.

## 4.5 Network-induced delay and its assignment

The chosen data-packet from the buffer of the data pre-processing unit is not necessarily the one which is just received from the network, no mention that the case where no data-packet is received. Thus, neither the network-transmission delay nor the data dropout is enough to describe the characteristic of the chosen data-packet. Instead the network-induced delay which is defined as follows is employed to do this job.

**Definition 4.1** *The network-induced delay $\tau_{ca}(k)$ (or $\tau_{sc}(k)$) denotes the difference between the current sampling instant $k$ and the time-stamp of the being used data-packet in the actuator (or in the controller).*

**Remark 4.1** *Choosing a data-packet from the data pre-processing buffer is equivalent to assigning a network-induced delay for the system.*

**Remark 4.2** *If a data-packet is used as soon as it is received (referring to Figure 4.2), then $\tau_{ca}(k) = \sigma_{ca}(k)$ or $\tau_{sc}(k) = \sigma_{sc}(k)$.*

Theoretically, the data pre-processing buffer can store data-packets received a long time ago, but these packets might result in too large network-induced delays if they are chosen to use in the controller or actuator. The least maximum network-induced delay can be determined from an extreme case where

1. the network-transmission delay of the control-to-actuator channel is constant of $\bar{\sigma}_{ca}$ (the upper bound);
2. a data-packet is received at the actuator node at the sampling instant $k$; and
3. data receiving fails at $k+i$, $i = 1, 2, \cdots, \varsigma_{ca}$, due to data dropouts.

In order to acquire the minimal network-induced delay, it is clear that at $k$ the just received data-packet is chosen such that $\tau_{ca}(k) = \sigma_{ca}(k) = \bar{\sigma}_{ca}$ and this packet is continuously chosen at $k+i$, $i = 1, 2, \cdots, \varsigma_{ca}$ such that $\tau_{ca}(k+i) = \tau_{ca}(k) + i$. Thus the least upper bound (LUB) of the network-induced delay is determined by

$$\bar{\tau}_{ca} = \bar{\sigma}_{ca} + \varsigma_{ca}. \tag{4.3}$$

Similarly, the LUB of the network-induced delay from the sensor to the controller is determined as

$$\bar{\tau}_{sc} = \bar{\sigma}_{sc} + \varsigma_{sc}. \tag{4.4}$$

It can be seen from (4.3) and (4.4) that the network-induced delay combines the impact of the network-transmission delay and the data dropout on the system performance.

With the LUB $\bar{\tau}_{ca}$ (or $\bar{\tau}_{sc}$) of the network-induced delay, it is unnecessary for the data pre-processing buffer to store data-packets older than $k-\bar{\tau}_{ca}$ (or $k-\bar{\tau}_{sc}$), that is to say, the buffer size is of $\bar{\tau}_{ca}+1$ (or $\bar{\tau}_{sc}+1$). Next, the available data-packets in the buffer are discussed as follows. At the actuator node, data-packets sent from the controller node between $k-\bar{\tau}_{ca}$ and $k-\bar{\sigma}_{ca}$, $u(k-\bar{\tau}_{ca}), u(k-(\bar{\tau}_{ca}-1)), \cdots, u(k-\bar{\sigma}_{ca})$, are definitely available if they have not dropped out. While packets $u(k-(\bar{\sigma}_{ca}-1)), u(k-(\bar{\sigma}_{ca}-2)), \cdots, u(k)$ are available if they have not dropped out and have network-transmission delays not greater than $(\bar{\sigma}_{ca}-1), (\bar{\sigma}_{ca}-2), \cdots, 0$, respectively. On the other hand, because consecutive data-packet dropouts are not greater than $\varsigma_{ca}=\bar{\tau}_{ca}-\bar{\sigma}_{ca}$, then at least one packet among $u(k-\bar{\tau}_{ca}), u(k-(\bar{\tau}_{ca}-1)), \cdots, u(k)$ is available. Thus, the network-induced delay $\bar{\tau}_{ca}(k)$ could be assigned within $[0, \bar{\tau}_{ca}]$ by choosing the corresponding packet in the buffer. It is worth noting that this delay-assignment might not be arbitrary because not all $\bar{\tau}_{ca}+1$ packets are definitely available. Hence, it is called "partly assigning". Again, the data pre-processing buffer in the controller node has similar results.

**Remark 4.3** *It is determined that $\tau_{ca}(k)$ or $\tau_{sc}(k)$ can be assigned to the interval $[\bar{\sigma}_{ca}, \bar{\tau}_{ca}]$ or $[\bar{\sigma}_{sc}, \bar{\tau}_{sc}]$, respectively.*

**Remark 4.4** *Without considering data dropout, $u(k-\bar{\sigma}_{ca})$ or $y(k-\bar{\sigma}_{sc})$ is definitely available in the buffer. If it is always chosen, the network-induced delay becomes constant, $\bar{\tau}_{ca}(k)=\bar{\sigma}_{ca}$ or $\bar{\tau}_{sc}(k)=\bar{\sigma}_{sc}$.*

Now, the problem is how to choose a proper data-packet from the data pre-processing buffer, or how to assign a network-induced delay, which will be addressed in Section 4.8 according to the stability analysis of switching systems. In most literatures of NCSs, the following two strategies guarantee the latest data-packet being always used.

- If there is data-packet(s) received, the newly received data-packet(s) is not necessary the latest data, so the data-packet with the smallest transmission delay, i.e., with the biggest time-stamp, is always be chosen. The data-packet used in the previous sampling instant is also compared.

- When there is no packet received, continue using the previous data packet.

In this method, the data pre-processing buffer size is 1, i.e., only the data-packet used in the previous sampling instant is stored. Correspondingly, the network-induced delay satisfies

$$\tau_{ca}(k) \leq \tau_{ca}(k-1)+1 \text{ or } \tau_{sc}(k) \leq \tau_{sc}(k-1)+1,$$

which means the current delay cannot increase more than 1 step compared to the previous one while it can decrease more than 1 step. However, this change is totally random, i.e., cannot be assigned.

## 4.6 Networked predictive control

For the sake of making this chapter self-contained, the networked predictive control method is summarized here; see Liu et al. (2006, 2007b) for details.

A typical networked predictive control system is described in Figure 4.3. The plant is modeled in discrete-time state space form

$$\begin{aligned} x(k+1) &= Ax(k)+Bu(k) \\ y(k) &= Cx(k) \end{aligned} \tag{4.5}$$

where $x(k) \in \mathbb{R}^n$, $u(k) \in \mathbb{R}^m$ and $y(k) \in \mathbb{R}^l$ are the state, the control input, the disturbance and the plant output, respectively; $A$, $B$ and $C$ are matrices of appropriate dimensions; $r(k)$ is the reference. The feedback signal can be either $y(k)$ (output feedback) or $x(k)$ (state feedback). For the single-input-single-output (SISO) plant, its input-output model is also given in the following discrete-time transfer function form

$$A(z^{-1})y(k) = B(z^{-1})u(k-1) \tag{4.6}$$

where $A(z^{-1}) \in \mathbb{R}(z^{-1}, n_a)$ and $B(z^{-1}) \in \mathbb{R}(z^{-1}, n_b)$ are polynomials

$$A(z^{-1}) = a_0 + a_1 z^{-1} + \cdots + a_{n_a} z^{-n_a}, \; a_0 = 1$$

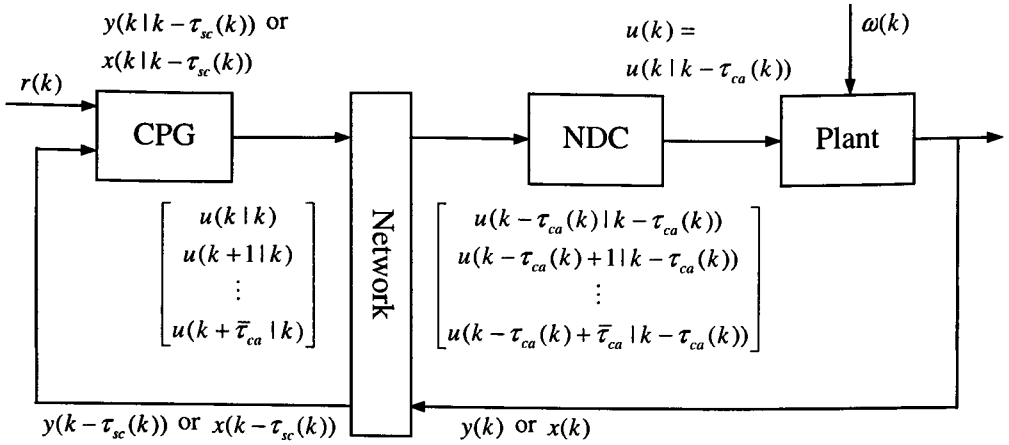



Figure 4.3: Typical structure of NPCSs.

$$B(z^{-1}) = b_0 + b_1 z^{-1} + \cdots + b_{n_b} z^{-n_b}$$

respectively.

In this chapter, $\mathbb{R}(z^{-1}, n)$ denotes the set of polynomials w.r.t $z^{-1}$ with order $n$ and with coefficients in $\mathbb{R}$. For $k_1 > k_2$, $y(k_1|k_2)$ (or $x(k_1|k_2)$, $u(k_1|k_2)$) denotes the prediction of $y(k_1)$ based on $y(k_2)$. $z^{-1}$ can be treated as either the complex variable or the shift operator in context. For the latter, it is defined that $z^{-1}y(k_1|k_2) = y(k_1 - 1|k_2)$. For the sake of consistency, $y(k)$ is sometime expressed as $y(k|k)$, thus $z^{-1}y(k|k) = y(k-1|k-1)$.

### 4.6.1 Predictive control in networked control systems

The network-induced delay makes the plant feedback signal and the control signal act on the controller and the actuator after the feedback network-induced delay $\tau_{sc}(k)$ and the forward channel network-induced delay $\tau_{ca}(k)$, respectively. In order to compensate for the impact of the network-induced delays on the system, networked predictive control scheme is employed. It includes two parts: control prediction generator (CPG) and network-induced delay compensator (NDC). Note that in this way, the control scheme is distributed at both sides of the network. The main control and calculation is in the CPG, while the NDC just does some simple operation. According to the plant model and some control law, CPG generates the plant output prediction $y(k|k - \tau_{sc}(k))$ or the state prediction $x(k|k - \tau_{sc}(k))$ to compensate for

$\tau_{sc}(k)$. The control input prediction $u(k+\tau_{ca}(k)|k)$ is also generated in CPG and sent to the plant side to compensate for $\tau_{ca}(k)$. However, $\tau_{ca}(k)$ is unknown until the control prediction is used in the actuator. Taking the advantage that data transmission in the network is in the form of network packet, a sequence of control prediction

$$U(k|k) = \begin{bmatrix} u(k|k) \\ u(k+1|k) \\ \vdots \\ u(k+\bar{\tau}_{ca}|k) \end{bmatrix} \tag{4.7}$$

is generated and sent to the actuator node in one packet. At the actuator node, NDC firstly determines $\tau_{ca}(k)$ of the to-be used control prediction sequence which implies this sequence is sent at $k-\tau_{ca}(k)$, i.e.,

$$U(k-\tau_{ca}(k)|k-\tau_{ca}(k)) = \begin{bmatrix} u(k-\tau_{ca}(k)|k-\tau_{ca}(k)) \\ u(k-\tau_{ca}(k)+1|k-\tau_{ca}(k)) \\ \vdots \\ u(k|k-\tau_{ca}(k)) \\ \vdots \\ u(k-\tau_{ca}(k)+\bar{\tau}_{ca}|k-\tau_{ca}(k)) \end{bmatrix}.$$

Then the $\tau_{ca}(k)+1$-th element in $U(k-\tau_{ca}(k)-\tau_{sc}(k))$ is selected as the actual control input which is

$$u(k) = u(k|-\tau_{ca}(k)). \tag{4.8}$$

In the following sections the generation of the control prediction sequence (4.7) according to the plant state-space model and the input-output model is respectively discussed.

### 4.6.2 Prediction based on the state-space model

The NPC method is proposed to compensate for the network-induced delay, so it is reasonable and simple to generate the control predictions based on a control law which is designed without considering the network.

According to model (4.5), the predictions of future system states up to $k$ based on $x(k-\tau_{sc}(k))$ are constructed as

$$\begin{aligned} x(k-\tau_{sc}(k)+1|k-\tau_{sc}(k)) &= Ax(k-\tau_{sc}(k)|k-\tau_{sc}(k)) \\ &\quad +Bu(k-\tau_{sc}(k)|k-\tau_{sc}(k)) \\ x(k-\tau_{sc}(k)+2|k-\tau_{sc}(k)) &= Ax(k-\tau_{sc}(k)+1|k-\tau_{sc}(k)) \\ &\quad +Bu(k-\tau_{sc}(k)+1|k-\tau_{sc}(k)) \\ &\vdots \\ x(k|k-\tau_{sc}(k)) &= Ax(k-1|k-\tau_{sc}(k)) \\ &\quad +Bu(k-1|k-\tau_{sc}(k)) \end{aligned} \tag{4.9}$$

where $x(k-\tau_{sc}(k)|k-\tau_{sc}(k)) = x(k-\tau_{sc}(k))$. Furthermore,

$$\begin{aligned} u(k-\tau_{sc}(k)|k-\tau_{sc}(k)) &= Kx(k-\tau_{sc}(k)|k-\tau_{sc}(k)) \\ u(k-\tau_{sc}(k)+1|k-\tau_{sc}(k)) &= Kx(k-\tau_{sc}(k)+1|k-\tau_{sc}(k)) \\ &\vdots \\ u(k-1|k-\tau_{sc}(k)) &= Kx(k-1|k-\tau_{sc}(k)) \end{aligned} \tag{4.10}$$

where gain $K \in R^{m\times n}$ is a state feedback controller designed according to the network-free system (i.e., the traditional local control system). Here the plant state $x(k)$ is assumed measurable, or otherwise a state observer is needed (Liu et al., 2007b). Combining (4.9) and (4.10) gives

$$x(k|k-\tau_{sc}(k)) = (A+BK)^{\tau_{sc}(k)}x(k-\tau_{sc}(k)). \tag{4.11}$$

Next, in the forward direction, based on $x(k|k-\tau_{sc}(k))$, similar steps to (4.9) and (4.10) are taken to obtain

$$\begin{aligned} x(k+1|k-\tau_{sc}(k)) &= Ax(k|k-\tau_{sc}(k))+Bu(k|k) \\ x(k+2|k-\tau_{sc}(k)) &= Ax(k+1|k-\tau_{sc}(k))+Bu(k+1|k) \end{aligned}$$

$$x(k+\bar{\tau}_{ca}|k-\tau_{sc}(k)) = Ax(k+\bar{\tau}_{ca}-1|k-\tau_{sc}(k))+Bu(k+\bar{\tau}_{ca}-1|k)$$

and the control prediction sequence

$$\begin{aligned}\begin{bmatrix} u(k|k) \\ u(k+1|k) \\ \vdots \\ u(k+\bar{\tau}_{ca}|k) \end{bmatrix} &= \begin{bmatrix} K \\ K(A+BK) \\ \vdots \\ K(A+BK)^{\bar{\tau}_{ca}} \end{bmatrix} x(k|k-\tau_{sc}(k)) \\ &= \begin{bmatrix} K(A+BK)^{\tau_{sc}(k)} \\ K(A+BK)^{\tau_{sc}(k)+1} \\ \vdots \\ K(A+BK)^{\tau_{sc}(k)+\bar{\tau}_{ca}} \end{bmatrix} x(k-\tau_{sc}(k)) \end{aligned} \tag{4.12}$$

It is clear that every control prediction in (4.12) is still in the form of static state-feedback control, but is nonlinear w.r.t. the original controller $K$. Later sections show that this makes the stability analysis and controller design difficult.

### 4.6.3 Prediction based on the polynomial model

A network-free controller for (4.6) is designed as

$$C(z^{-1})u(k) = D(z^{-1})(r(k)-y(k)) \tag{4.13}$$

where $C(z^{-1}) \in \mathbb{R}(z^{-1}, n_c)$ with $c_0 = 1$ and $D(z^{-1}) \in \mathbb{R}(z^{-1}, n_d)$.

Use the following Diophantine equations

$$A(z^{-1})E_i(z^{-1}) + z^{-i-\tau_{sc}(k)}F_i(z^{-1}) = 1 \tag{4.14}$$

where $i = 1, 2, \cdots, \bar{\tau}_{ca}$, $E_i(z^{-1}) \in \mathbb{R}(z^{-1}, i+\tau_{sc}(k)-1)$ and $F_i(z^{-1}) \in \mathbb{R}(z^{-1}, n_a-1)$, and the plant model (4.6) to produce

$$\begin{aligned} y(k+1|k) &= F_1(z^{-1})y(k-\tau_{sc}(k)) + B(z^{-1})E_1(z^{-1})u(k|k) \\ y(k+2|k) &= F_2(z^{-1})y(k-\tau_{sc}(k)) + B(z^{-1})E_2(z^{-1})u(k+1|k) \end{aligned} \tag{4.15}$$

$$\vdots$$

$$y(k+\bar{\tau}_{ca}+1|k) = F_{\bar{\tau}_{ca}}(z^{-1})y(k-\tau_{sc}(k)) + B(z^{-1})E_{\bar{\tau}_{ca}}(z^{-1})u(k+\bar{\tau}_{ca}|k)$$

Combine (4.13) and (4.15) to get

$$\begin{aligned}
C(z^{-1})u(k|k) &= D(z^{-1})r(k) - D(z^{-1})F_1(z^{-1})y(k-\tau_{sc}) \\
&\quad -D(z^{-1})B(z^{-1})E_1(z^{-1})u(k|k) \\
C(z^{-1})u(k+1|k) &= D(z^{-1})r(k+1) - D(z^{-1})F_2(z^{-1})y(k-\tau_{sc}) \\
&\quad -D(z^{-1})B(z^{-1})E_2(z^{-1})u(k+1|k) \\
&\vdots \\
C(z^{-1})u(k+\bar{\tau}_{ca}|k) &= D(z^{-1})r(k+\bar{\tau}_{ca}+1) - D(z^{-1})F_{\bar{\tau}_{ca}}(z^{-1})y(k-\tau_{sc}) \\
&\quad -D(z^{-1})B(z^{-1})E_{\bar{\tau}_{ca}}(z^{-1})u(k+\bar{\tau}_{ca}|k).
\end{aligned}$$

By splitting

$$\begin{bmatrix} C(z^{-1})u(k|k) \\ C(z^{-1})u(k+1|k) \\ \vdots \\ C(z^{-1})u(k+\bar{\tau}_{ca}|k) \end{bmatrix} = H(z^{-1})u(k-1|k-1) + LU(k|k)$$

and

$$\begin{bmatrix} D(z^{-1})B(z^{-1})E_1(z^{-1})u(k|k) \\ D(z^{-1})B(z^{-1})E_2(z^{-1})u(k+1|k) \\ \\ D(z^{-1})B(z^{-1})E_{\bar{\tau}_{ca}}(z^{-1})u(k+\bar{\tau}_{ca}|k) \end{bmatrix} = \Gamma(z^{-1})u(k-1|k-1) + MU(k|k),$$

control prediction sequence is obtained

$$U(k|k) = P(z^{-1})r(k+\bar{\tau}_{ca}+1) - Q(z^{-1})y(k-\tau_{sc}(k)) - S(z^{-1})u(k-1|k-1) \quad (4.16)$$

where

$$\begin{aligned} T(z^{-1}) &= \left[ \begin{array}{cccc} T_0(z^{-1}) & T_1(z^{-1}) & \cdots & T_{\bar{\tau}_{ca}}(z^{-1}) \end{array} \right]^{\mathrm{T}} \\ &= (L+M)^{-1} \left[ \begin{array}{cccc} z^{-\bar{\tau}_{ca}} & z^{-\bar{\tau}_{ca}+1} & \cdots & 1 \end{array} \right]^{\mathrm{T}} D(z^{-1}) \end{aligned}$$

$$\begin{aligned} Q(z^{-1}) &= \left[ \begin{array}{cccc} Q_0(z^{-1}) & Q_1(z^{-1}) & \cdots & Q_{\bar{\tau}_{ca}}(z^{-1}) \end{array} \right]^{\mathrm{T}} \\ &= (L+M)^{-1} F(z^{-1}) D(z^{-1}) \end{aligned}$$

$$\begin{aligned} S(z^{-1}) &= \left[ \begin{array}{cccc} S_0(z^{-1}) & S_1(z^{-1}) & \cdots & S_{\bar{\tau}_{ca}}(z^{-1}) \end{array} \right]^{\mathrm{T}} \\ &= (L+M)^{-1} \left( \Gamma(z^{-1}) + H(z^{-1}) \right) \end{aligned}$$

$$F(z^{-1}) = \left[ \begin{array}{cccc} F_1(z^{-1}) & F_2(z^{-1}) & \cdots & F_{1+\bar{\tau}_{ca}}(z^{-1}) \end{array} \right]^{\mathrm{T}}$$

$$E(z^{-1}) = \left[ \begin{array}{cccc} E_1(z^{-1}) & E_2(z^{-1}) & \cdots & E_{1+\bar{\tau}_{ca}}(z^{-1}) \end{array} \right]^{\mathrm{T}}$$

$$\begin{aligned} G(z^{-1}) &= \left[ \begin{array}{cccc} G_0(z^{-1}) & G_1(z^{-1}) & \cdots & G_{\bar{\tau}_{ca}}(z^{-1}) \end{array} \right]^{\mathrm{T}} \\ &= D(z^{-1}) B(z^{-1}) E(z^{-1}) \end{aligned}$$

$$\begin{aligned} \Gamma(z^{-1}) &= \left[ \begin{array}{cccc} \Gamma_0(z^{-1}) & \Gamma_1(z^{-1}) & \cdots & \Gamma_{\bar{\tau}_{ca}}(z^{-1}) \end{array} \right]^{\mathrm{T}} \\ &= \left[ \begin{array}{cccc} G_0(z^{-1}) - g_{00} & G_1(z^{-1}) - g_{10} - g_{11} z^{-1} & \cdots & G_{\bar{\tau}_{ca}}(z^{-1}) - \sum_{i=0}^{\bar{\tau}_{ca}} g_{\bar{\tau}_{ca} i} z^{-i} \end{array} \right]^{\mathrm{T}} \end{aligned}$$

$$\begin{aligned} H(z^{-1}) &= \left[ \begin{array}{cccc} H_0(z^{-1}) & H_1(z^{-1}) & \cdots & H_{\bar{\tau}_{ca}}(z^{-1}) \end{array} \right]^{\mathrm{T}} \\ &= \left[ \begin{array}{cccc} C(z^{-1}) - c_0 & C(z^{-1}) - c_0 - c_1 z^{-1} & \cdots & C(z^{-1}) - \sum_{i=0}^{\bar{\tau}_{ca}} c_i z^{-i} \end{array} \right]^{\mathrm{T}} \end{aligned}$$

$$M = \begin{bmatrix} g_{00} & 0 & \cdots & 0 \\ g_{11} & g_{10} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ g_{\bar{\tau}_{ca}\bar{\tau}_{ca}} & g_{\bar{\tau}_{ca}(\bar{\tau}_{ca}-1)} & \cdots & g_{\bar{\tau}_{ca}0} \end{bmatrix}$$

$$L = \begin{bmatrix} c_0 & 0 & \cdots & 0 \\ c_1 & c_0 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ c_{\bar{\tau}_{ca}} & c_{\bar{\tau}_{ca}-1} & \cdots & c_0 \end{bmatrix}.$$

**Remark 4.5** *In the NPC method, the control prediction sequence is generated according to a known controller, which is different from traditional predictive control methods, such as GPC. NPC focuses on the strategy of compensating for the network-induced delay, rather than the controller design, whereas GPC emphasizes the controller design subject to the optimization of some performance indexes. There is no conflict that both GPC and NPC are applied to networked control systems (Liu et al., 2006).*

## 4.7 Stability analysis of NPCSs

In this section, only the state-space method is discussed. The polynomial method gives similar results.

### 4.7.1 Closed-loop system

In the state-space method, the closed-loop system is obtained as

$$x(k+1) = Ax(k) + M_{\tau(k)}x(k-\tau(k)) \tag{4.17}$$

by combining (4.5), (4.8) and (4.12), where

$$\tau(k) = \tau_{sc}(k) + \tau_{ca}(k)$$

$$M_{\tau(k)} = BK(A+BK)^{\tau(k)}$$

and $\tau(k) \in \{0, 1, \cdots, \bar{\tau}_{sc} + \bar{\tau}_{ca}\}$ is the round-trip network-induced delay. Let $\bar{\tau} = \bar{\tau}_{sc} + \bar{\tau}_{ca}$. By augmenting the system states, (4.17) is equivalent to

$$X(k+1) = \Lambda_{\tau(k)} X(k) \tag{4.18}$$

where

$$X(k) = \begin{bmatrix} x(k) \\ x(k-1) \\ \vdots \\ x(k-\tau(k)) \\ \vdots \\ x(k-\bar{\tau}) \end{bmatrix}, \quad \Lambda_{\tau(k)} = \begin{bmatrix} A & & & M_{\tau(k)} & & \\ I & 0 & & & & \\ & \ddots & \ddots & & & \\ & & I & 0 & & \\ & & & \ddots & \ddots & \\ & & & & I & 0 \end{bmatrix}.$$

In $\Lambda_{\tau(k)}$, for $\tau(k) \neq 0$, $M_{\tau(k)}$ is at the position $(1,\ \tau(k)+1)$. Note that when $\tau(k) = 0$,

$$\Lambda_0 = \begin{bmatrix} A+BK & & & & & \\ I & 0 & & & & \\ & \ddots & \ddots & & & \\ & & I & 0 & & \\ & & & \ddots & \ddots & \\ & & & & I & 0 \end{bmatrix}$$

is equivalent to the network-free system.

### 4.7.2 Further understanding of fixed network-induced delay

Networked control systems with constant network-induced delay do exist. For example, some fieldbus or Intranet NCSs can be modeled as constant delay systems because the network is exclusively used. For any constant $\tau(k) \equiv \tau$ in $\{0, 1, \cdots, \bar{\tau}\}$, (4.18) is stable if and only if all eigenvalues of $\Lambda_\tau$ are within the unit circle. This is

equivalent to that all eigenvalues of

$$\tilde{\Lambda}_\tau = \begin{bmatrix} A & & & M_\tau \\ I & 0 & & \\ & \ddots & \ddots & \\ & & I & 0 \end{bmatrix} \in R^{\bar{\tau}(\tau+1)\times\bar{\tau}(\tau+1)}$$

are within the unit circle. Without considering zero eigenvalues, the characteristic polynomial can be simplified as

$$\begin{aligned}
\det(zI-\tilde{\Lambda}_\tau) &= \det\left(\begin{bmatrix} zI-A & 0 & \cdots & 0 & -M_\tau \\ -I & zI & \cdots & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \cdots & zI & 0 \\ 0 & 0 & \cdots & -I & zI \end{bmatrix}\right) \\
&= \det\left(\begin{bmatrix} zI-A-z^{-\tau}M_\tau & -z^{-\tau+1}M_\tau & \cdots & -z^{-1}M & -M_\tau \\ 0 & zI & \cdots & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \cdots & zI & 0 \\ 0 & 0 & \cdots & 0 & zI \end{bmatrix}\right) \\
&= \det\left(\begin{bmatrix} zI & \cdots & 0 & 0 \\ \vdots & \ddots & \vdots & \vdots \\ 0 & \cdots & zI & 0 \\ 0 & \cdots & 0 & zI \end{bmatrix}\right)\det(zI-A-z^{-\tau}M_\tau) \\
&= \det(z^\tau(zI-A)-M_\tau) \\
&= \det(z^\tau(zI-A)-BK(A+BK)^\tau) \\
&= \det(z^\tau(zI-A)-(A+BK)^{\tau+1}+A(A+BK)^\tau) \\
&= \det(z^{\tau+1}I-(A+BK)^{\tau+1}-A(z^\tau I-(A+BK)^\tau) \\
&= \det((zI-A-BK)\times \\
&\quad \Sigma_{j=0}^{\tau} z^j(A+BK)^{\tau-j}-A(zI-A-BK)\Sigma_{j=0}^{\tau-1}z^j(A+BK)^{\tau-1-j}) \\
&= \det(zI-A-BK)\det(\Sigma_{j=0}^{\tau}z^j(A+BK)^{\tau-j}-A\Sigma_{j=0}^{\tau-1}z^j(A+BK)^{\tau-1-j})
\end{aligned}$$

$$
\begin{aligned}
&= \det(zI - A - BK)\det(\Sigma_{j=0}^{\tau} z^j (A+BK)^{\tau-j} - \\
&\quad (A+BK)\Sigma_{j=0}^{\tau-1} z^j (A+BK)^{\tau-1-j} + BK\Sigma_{j=0}^{\tau-1} z^j (A+BK)^{\tau-1-j}) \\
&= \det(zI - A - BK)\det(z^{\tau} I + BK\Sigma_{j=0}^{\tau-1} z^j (A+BK)^{\tau-1-j}).
\end{aligned}
$$

This means that stable $\Lambda_\tau$ implies $\Lambda_0$ is also stable, but can not guarantee $\Lambda_j$ ( $j = 0, 1, \cdots, \bar{\tau},\ j \neq \tau,\ j \neq 0$) is stable.

## 4.8 Delay assignment in NPCS

With random $\tau(k)$, (4.18) is a switched system

$$X(k+1) = \Lambda_{\delta(k)} X(k) \tag{4.19}$$

with switching signal

$$\delta(k) : \{0, 1, 2, \cdots\} \rightarrow \{0, 1, \cdots, \bar{\tau}\}.$$

Any value of round-trip network-induced delay corresponds to a subsystem.

**Lemma 4.1** *[from Liu et al. (2006)] System (4.19) is stable for any switching signal $\delta(k)$ if there exists a common positive definite matrix $P$ which satisfies the following linear matrix inequalities (LMIs)*

$$\Lambda_i^{\mathrm{T}} P \Lambda_i - P < 0,\ \forall i \in \psi. \tag{4.20}$$

### 4.8.1 Direct results from Lemma 4.1

In general, Lemma 4.1 is strict. It requires that not only all subsystems are stable but also a common $P$ exists for them. This strictness is due to the limitation of the above delay-compensation strategy and the randomness of the network-induced delay. At any time instant, the control input has only one random candidate. This means which subsystem in (4.19) is activated is completely determined by the network-induced delay. In other words, the switching signal is not designable unless we know

some statistical characteristics about the network (this is not the subject of this chapter). Therefore the stability analysis of system (4.19), such as Lemma 4.1, has to be subject to the arbitrary switching of all subsystems. As mentioned before, the delay is assignable. Now the problem is how to design the switching signal (assign the delay). For the system performance consideration, at any time instant, it is reasonable to always choose the subsystem which results in a small delay from all available ones because it reflects the latest information of the system. Above all, however, this must meet the system stability requirements.

**Assumption 4.3** *Subsystems* $\{\Lambda_{\bar{\sigma}}, \Lambda_{\bar{\sigma}+1}, \cdots, \Lambda_{\bar{\tau}}\}$ *in (4.19) have a common* $P$ *such that their corresponding LMIs in (4.20) hold.*

**Definition 4.2** *All subsystems of (4.19) are classified into the following four categories:*

$\Psi_1$: $\{\Lambda_{\bar{\sigma}}, \Lambda_{\bar{\sigma}+1}, \cdots, \Lambda_{\bar{\tau}}\}$;

$\Psi_2$: *Stable subsystems in* $\{\Lambda_0, \Lambda_1, \cdots, \Lambda_{\bar{\sigma}-1}\}$ *and they have a common* $P$ *with* $\Psi_1$;

$\Psi_3$: *Stable subsystems in* $\{\Lambda_0, \Lambda_1, \cdots, \Lambda_{\bar{\sigma}-1}\}$ *but they have no common* $P$ *with* $\Psi_1$;

$\Psi_4$: *Unstable subsystems in* $\{\Lambda_0, \Lambda_1, \cdots, \Lambda_{\bar{\sigma}-1}\}$.

*Correspondingly,* $\psi_1$, $\psi_2$, $\psi_3$ *and* $\psi_4$ *denote the subscript or delay sets to which the subsystems correspond.*

At time $k$, some or all control input candidates of $u(k|k-\bar{\sigma}), u(k|k-(\bar{\sigma}+1)), \cdots, u(k|k-\bar{\tau})$ are always available, so we can always assign the system into $\Psi_1$. The following corollary is straightforward from Lemma 4.1.

**Corollary 4.1** *Under Assumption 4.3, system (4.19) is stable with any switching signal*

$$\delta_1(k) : \{0, 1, \cdots\} \rightarrow \psi_1.$$

Compared to Lemma 4.1, Corollary 4.1 requires less subsystems having a common $P$, so it is easier to satisfy. On the other hand, for unstable subsystems $\Psi_4$,

it is reasonable to avoid assigning the system to any one of them. For subsystems $\Psi_2$ and $\Psi_3$, it is better to assign the system to them, if possible, rather than to $\Psi_1$ because $\Psi_2$ and $\Psi_3$ result in smaller delays. In more detail, for $\Psi_2$, we can similarly get the following corollary.

**Corollary 4.2** *Under the definition of $\Psi_2$, system (4.19) is stable with any switching signal*

$$\delta_2(k) : \{0, 1, \cdots\} \to \psi_1 \cup \psi_2.$$

### 4.8.2 Switching signal governed by average dwell time

First, amend Assumption 4.3 and Definition 4.2 as follows.

**Assumption 4.4** *Given $\alpha > 0$, subsystems $\Psi_{\alpha 1} = \{\Lambda_{\bar{\sigma}}, \Lambda_{\bar{\sigma}+1}, \cdots, \Lambda_{\bar{\tau}}\}$ have a common positive definite matrix $P$ such that the following LMIs*

$$\Lambda_i^{\mathrm{T}} P \Lambda_i - e^{-\alpha} P < 0 \tag{4.21}$$

*hold, where $\forall i \in \psi_{\alpha 1} = \{\bar{\sigma}, \bar{\sigma}+1, \cdots, \bar{\tau}\}$.*

**Definition 4.3** *$\Psi_{\alpha 1}$, $\Psi_{\alpha 2}$, $\Psi_{\alpha 3}$, $\Psi_{\alpha 4}$ and $\psi_{\alpha 1}$, $\psi_{\alpha 2}$, $\psi_{\alpha 3}$, $\psi_{\alpha 4}$ have similar meanings as Definition 4.2; Denote $P_{\bar{\sigma}}$ as the common matrix for $\Lambda_i$, $\forall i \in \psi_{\alpha 1} \cup \psi_{\alpha 2}$ and $P_j$ for $\Lambda_j$, $\forall j \in \psi_{\alpha 3}$.*

Now the problem is to study the stability of switched system (4.19) with switching signal

$$\delta_3(k) : \{0, 1, \cdots\} \to \psi_{\alpha 1} \cup \psi_{\alpha 2} \cup \psi_{\alpha 3}.$$

**Definition 4.4** *For switching signal $\sigma_3(k)$ and $k_2 > k_1 \geq 0$, let $N_{\sigma_3(k)}(k_1, k_2)$ denote the number of switching of $\sigma_3(k)$ over the time interval $[k_1, k_2]$. For given $t_a > 0$, $N_0 \geq 0$, if the inequality*

$$N_{\sigma_3(k)}(k_1,\ k_2) \leq N_0 + \frac{k_2 - k_1}{t_a} \tag{4.22}$$

*holds, then the positive constant $t_a$ is called average dwell time and $N_0$ is called chattering bound. For the sake of convenience, $N_0 = 0$ is chosen in this chapter.*

**Theorem 4.1** *Under Assumption 4.4 and Definition 4.3, system (4.19) is exponentially stable for switching signal $\sigma_3(k)$ with average dwell time*

$$t_a \geq t_a^* = \frac{\ln \mu}{\alpha} \tag{4.23}$$

*where constant $\mu \geq 1$ satisfies*

$$P_i \leq \mu P_j, \tag{4.24}$$

*$\forall i, j \in \psi_{\alpha 3} \cup \{\bar{\sigma}\}$. Moreover, the system state decay rate is given by*

$$\|X_k\| \leq \sqrt{\frac{b}{a}} e^{-\frac{1}{2}(\alpha - \frac{\ln \mu}{t_a})k} \|X_0\|$$

*where*

$$a = \min_i \underline{\lambda}(P_i)$$

$$b = \max_i \overline{\lambda}(P_i),$$

*$\underline{\lambda}(P_i)$ and $\overline{\lambda}(P_i)$ respectively denote the minimum and the maximum eigenvalues of $P_i$, $\forall i \in \psi_{\alpha 3} \cup \{\bar{\sigma}\}$.*

**Proof**: Choose a piecewise Lyapunov function candidate as

$$V_k = V_k^{(i)} = X_k^{\mathrm{T}} P_i X_k \tag{4.25}$$

when some subsystem is activated at time $k$, where $P_i$s, $\forall i \in \psi_{\alpha 3} \cup \{\bar{\sigma}\}$ are from Definition 4.3. Along the trajectory of system (4.19) for any Lyapunov function $V_k^{(i)}$ in (4.25),

$$\begin{aligned} V_{k+1}^{(i)} - e^{-\alpha} V_k^{(i)} &= X_{k+1}^{\mathrm{T}} P_i X_{k+1} - e^{-\alpha} X_k^{\mathrm{T}} P_i X_k \\ &= X_k^{\mathrm{T}} \Lambda_i^{\mathrm{T}} P_i \Lambda_i X_k - e^{-\alpha} X_k^{\mathrm{T}} P_i X_k \\ &= X_k^{\mathrm{T}} \left( \Lambda_i^{\mathrm{T}} P_i \Lambda_i - e^{-\alpha} P_i \right) X_k \\ &< 0. \end{aligned}$$

As a result, if there is no switching on the time interval $[0, k]$,

$$V_k^{(i)} \leq e^{-\alpha} V_{k-1}^{(i)} \leq \cdots \leq e^{-\alpha k} V_0^{(i)}. \tag{4.26}$$

On the other hand, according to (4.24) and (4.25), it is obtained that

$$V_k^{(i_1)} \leq \mu V_k^{(i_2)} \tag{4.27}$$

where $\forall i_1, i_2 \in \psi_{\alpha 3} \cup \{\bar{\sigma}\}$. At time $k$, let $0 = t_0 < k_1 < \cdots < t_q = t_{N_{\sigma_3(k)}}$ denote the switching time instants of $\sigma_3(k)$ over the interval $[0,\ k]$. From (4.26),

$$V_k \leq e^{-\alpha(k-t_q)} V_{t_q}^{(\sigma_3(t_q))} \tag{4.28}$$

In discrete-time systems, Lyapunov function keeps its value during the sample period and jumps on the time instant. Combining (4.27) and (4.28) leads to

$$\begin{aligned} V_k &\leq e^{-\alpha(k-t_q)} V_{t_q}^{(\sigma_3(t_q))} \\ &\leq e^{-\alpha(k-t_q)} \mu V_{t_q^-}^{(\sigma_3(t_q^-))} \\ &\leq e^{-\alpha(k-t_q)} \mu e^{-\alpha(t_q - t_{q-1})} V_{t_{q-1}}^{(\sigma_3(t_{q-1}))} \\ &= e^{-\alpha(k-t_{q-1})} \mu V_{t_{q-1}}^{(\sigma_3(t_{q-1}))} \\ &\vdots \\ &\leq e^{-\alpha k} \mu^{N_{\sigma_3(k)}(0,\ k)} V_0 \end{aligned}$$

where $t_q^-$ denotes the left-side limit of $t_q$. In view of (4.22), $V_k$ is further expressed as

$$\begin{aligned} V_k &\leq e^{-\alpha k} \mu^{N_{\sigma_3(k)}(0,\ k)} V_0 \\ &\leq e^{-(\alpha - \frac{\ln \mu}{t_a})k} V_0 \end{aligned}$$

On the other hand, from Lyapunov function (4.25), there are

$$\underline{\lambda}(P_i) \|X_k\|^2 \leq V_k$$

$$V_0 \leq \overline{\lambda}(P_i) \|X_0\|^2$$

Furthermore,

$$\|X_k\|^2 \leq \frac{1}{a} e^{-(\alpha - \frac{\ln \mu}{t_a})} V_0 \leq \frac{b}{a} e^{-(\alpha - \frac{\ln \mu}{t_a})} \|X_0\|^2$$

$$\|X_k\| \leq \sqrt{\frac{b}{a}} e^{-\frac{1}{2}(\alpha - \frac{\ln \mu}{t_a})k} \|X_0\| .$$

When $\alpha - \frac{\ln \mu}{t_a} > 0$, $t_a > \frac{\ln \mu}{\alpha}$, $\|X_k\|$ is exponential decaying. This completes the proof. □

Because subsystems $\Psi_{\alpha 1} \cup \Psi_{\alpha 2}$ have a common Lyapunov function, the switching among them is not included in the system switching number as far as the average dwell time is concerned. This is classified as "noneffective switching", compared with "effective switching" which takes place among $\Psi_{\alpha 3}$ or between $\Psi_{\alpha 3}$ and $\Psi_{\alpha 1} \cup \Psi_{\alpha 2}$. Bear in mind that the system can always be assigned into $\Psi_{\alpha 1} \cup \Psi_{\alpha 2}$ ($\Psi_{\alpha 1}$, more accurate to say). When effective switching happens frequently, let the system dwell in $\Psi_{\alpha 1} \cup \Psi_{\alpha 2}$ more often to guarantee the average dwell time $t_a$ satisfying (4.23). In addition, if some subsystems of $\Psi_{\alpha 3}$ have a common Lyapunov function like (4.25), they can be classified into a subcategory and the switching among them is also noneffective.

**Remark 4.6** *The constant $\mu \geq 1$ definitely exists. For example, $\mu = \frac{b}{a}$ is feasible for (4.24), though it might be big. In general, we can find a much smaller value than that.*

**Remark 4.7** *$\alpha = 0$ implies the average dwell time $\infty$, i.e., no switching is allowed (the system has to dwell in $\Psi_{\alpha 1} \cup \Psi_{\alpha 2}$). This is the case of Corollary 4.2. On the other hand, if $\mu = 1$ is feasible, i.e., all $P_i$s are the same, then system (4.19) is exponentially stable with any switching signal $\sigma_3(k)$. This is the exponential stability version of Corollary 4.2.*

### 4.8.3 Implementation algorithm

For a general switched system, all of its subsystems are available at any time instant, so the switching signal can be designed before the system is implemented. In this

**Algorithm 2** Implementation of the proposed network-induced delay compensation strategy with the switching signal governed by the average dwell time

*Inputs:* plant $A$, $B$; controller $K$ to make $(A, B)$ stable; network-transmission delay upper bound $\bar{\sigma}_{ca}$ and $\bar{\sigma}_{sc}$ and maximum consecutive data dropouts $\varsigma_{ca}$ and $\varsigma_{sc}$ in controller-to-actuator and sensor-to-controller network channel, respectively.

1. Calculate all subsystems in (4.19). Given $\alpha \geq 0$, classify them into $\Psi_{\alpha 1}$, $\Psi_{\alpha 2}$, $\Psi_{\alpha 3}$ and $\Psi_{\alpha 4}$ according to their definitions and get $P_i$s, $\forall i \in \psi_{\alpha 3} \cup \{\bar{\tau}\}$;
2. Calculate $\mu$ and $t_a^*$ from (4.23) and (4.24);
3. Loop $k$
   (a) At controller node, calculate control prediction sequence $U(k|k)$ and send it with the time-stamp;
   (b) At actuator node, receive control prediction sequence and update the buffer which stores all available sequences according to the time-stamps;
   (c) From all available control input candidates, check the one resulting in a smaller delay. If its corresponding subsystem is in $\Psi_{\alpha 4}$, repeat this step;
   (d) If the subsystem is in $\Psi_{\alpha 3}$, check the switching number $N_{\sigma_3}(k)$ supposing this candidate is used as the control input. If (4.22) is not violated, use this candidate and update $N_{\sigma_3}(k)$; otherwise go back to (c);
   (e) If the subsystem is in $\Psi_{\alpha 1} \cup \Psi_{\alpha 2}$, use this candidate.

chapter, although the improved delay-compensation strategy makes it possible to design the switching signal for system (4.19), which subsystems are available at each time instant is still uncertain. Therefore the online design of the switching signal is needed. That is, at time $k$, depending on the total switching numbers before time $k$, make sure that the current selection of some subsystem will not violate (4.22). In practice, considering the switching number in a time interval periodically instead of the whole time horizon is easier to handle.

In the end, in order to make it clear how to implement the proposed delay-compensation strategy and the average dwell time method in NPCSs, an algorithm is summarized in Algorithm 2.

## 4.9 Example

Consider a NPCS where the network only exists in the forward channel, and the random network-transmission delay with upper bound $\bar{\sigma}_{ca} = 3$ and the consecutive data dropout is no greater than 1, so the control prediction horizon $\bar{\tau}_{ca} = 4$. For an integral plant

$$\frac{B(z^{-1})}{A(z^{-1})} = \frac{z^{-1}}{1 - z^{-1}},$$

a PI controller to make the network-free system stable is designed as

$$\frac{D(z^{-1})}{C(z^{-1})} = \frac{1.18 - 0.8z^{-1}}{1 - z^{-1}}.$$

In all subsystems, $\Lambda_2$ is unstable and $\Lambda_0$, $\Lambda_1$, $\Lambda_3$, $\Lambda_4$ are stable. A common $P$ for $\Lambda_0$, $\Lambda_1$, $\Lambda_3$, $\Lambda_4$ to satisfy the corresponding LMIs in (4.20) can be found using MATLAB/LMI Toolbox. Therefore, according to Corollary 4.2, the system is stable if the control input candidate $u(k|k-2)$ is not used whenever it is available.

Given $\alpha = 0.08$, Assumption 4.4 is satisfied and $P_3$ is solved. $\Lambda_0$ or $\Lambda_1$ has no common $P$ with $\Lambda_3$ and $\Lambda_4$ to satisfy LMIs in the form of (4.21), while $\Lambda_0$ and $\Lambda_1$ themselves share a common $P$ (denoted as $P_0$). This means that effective switching only happens when system switches from $\Lambda_0$ or $\Lambda_1$ to $\Lambda_3$ or $\Lambda_4$, and vice versa. From $P_0$ and $P_3$, $a = 10.5253$, $b = 2.5767 \times 10^3$ and $\mu = 1.3$ is feasible for (4.24). By (4.23), average dwell time $t_a^* = 3.2796$. According to Theory 4.1, when taking $t_a = 5 \geq t_a^*$, the state decay is given as $\|X_k\| \leq 15.6464e^{-0.0138t} \|X_0\|$. If the time interval is set to 20 steps, then during this interval the switching number is not allowed to be more than 4 according to (4.22).

Simulation results are shown in Figure 4.4, where the system is tracking a unit step reference . It can be seen that the system is unstable under arbitrary switching signal for all subsystems. When subsystem $\Lambda_2$ is ruled out, the system is stable and Theorem 4.1 gives a better response than Corollary 4.2.

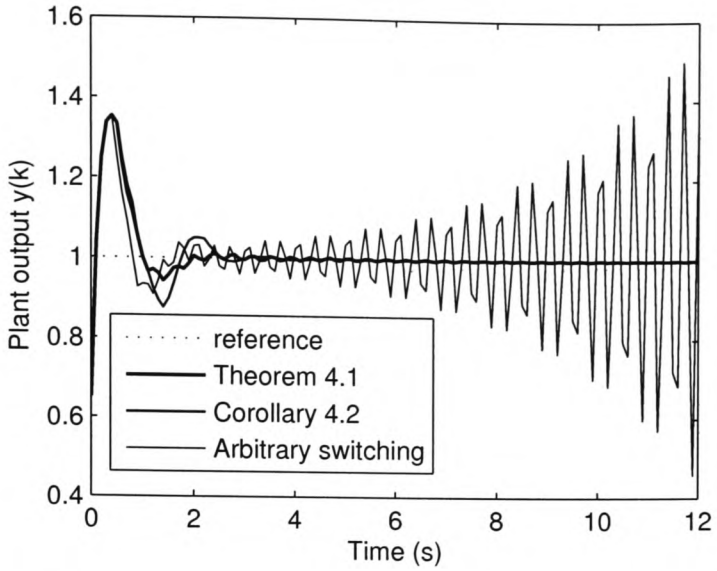



Figure 4.4: System response comparison under different switching signals.

## 4.10 Summary

A NPCS with random network-induced delay has been discussed as a switched system. In the original delay-compensation method, the switching signal is not designable which results that all subsystems have to be stable and that they share a common Lyapunov function. This chapter has improved the compensation strategy so that we can assign the subsystem or design the switching signal as we want. The improved NPC method works even if there exists unstable subsystems because the method is designed not to activate theses subsystems. As for the remaining stable subsystems, by applying the average dwell time method, a common Lyapunov function is not necessary for them. When the switching signal is designed to meet the average dwell time requirement, the system is exponentially stable.

# Chapter 5

# Networked Predictive Control of Magnetic Levitation System

This chapter discusses the control of a magnetic levitation (MagLev) system over networks. Firstly, a test-rig is set up to implement control. Feedback linearization and direct local linearization methods for the nonlinear MagLev system are presented. In order to improve the control performance, the networked predictive control method is employed, where the system model is identified in real-time. Both local control and networked control are implemented on this test-rig. Networked predictive control demonstrates a clear performance advantage over other networked control strategies which do not incorporate compensation for the network-induced delay.

## 5.1 Introduction

Magnetic levitation (MagLev) system has been widely studied and applied to high-speed transportation, magnetic bearing, vibration isolation and many other applications (Downer, 1980; Dahlen, 1985; Dussaux, 1990; Limbert et al., 1990; Yamamura and Yamaguchi, 1990), due to its advantage of frictionless contact.

Magnetic levitation system is typically open-loop unstable, time-varying and highly nonlinear, so it presents significant control challenges. Generally, most control methods are based on a linearized model of the MagLev system at a nominal operating point (Downer, 1980; Barie and Chiasson, 1996). The tracking performance of the resulting closed-loop system, however, deteriorates rapidly with increasing deviations from the nominal operating point. Two main approaches to improve the tracking performance have been reported in the literature. One is that of gain scheduling where the nonlinearity of the MagLev system is successively linearized at various operating points with a suitable controller designed for each of these operating points. Gain scheduling controllers (Kim and Kim, 1994) require the operating range to be broken up into very fine intervals and stored in a look-up table of controller gains. The other approach is feedback linearization (Barie and Chiasson, 1996; Charara et al., 1996; Joo and Seo, 1997; Trumpler et al., 1997; Hajjaji and Ouladsine, 2001), which utilizes the nonlinear description of the system and hence yields consistent performance largely independent of operating points. Other control methods, like sliding mode control (Cho et al., 1993), back-stepping control (Lin et al., 2007) and dynamic surface control (Yang et al., 2004), were also proposed to actively tackle system uncertainties and robustness. As for the issue of the model identification of MagLev system, some results can be found in Sun et al. (1999); Yang et al. (2002b).

Control of a MagLev system over a network presents another level of challenge. Network-induced transmission delay and data dropout in the closed-loop makes it more difficult for the system to achieve the desired performance, even to stabilize the system is a particular challenge. Because MagLev system is a fast dynamic system, control of it requires a small sampling period. This means the resulting network-induced delay might be large w.r.t. the sampling period. Networked control of

MagLev system over a LAN has been reported by Kim et al. (2006a,b), where an auto regressive prediction model is built to compensate for the network-induced delay.

In this chapter, several control methods based on the local linearization model and feedback linearization model of the plant and their networked predictive control versions are studied for a MagLev prototype system which was manufactured by BYTRONIC company. In order to implement these control strategies, a MagLev test-rig including MagLev system, data acquisition and networked controller is set up.

## 5.2 MagLev test rig

The MagLev test rig studied in this chapter is set up as shown in Figure 5.1. The MagLev system is connected to the network via a NetConController and controlled by a remote controller. This forms a networked control system. NetConController, NetConLink and NetConTop compose an embedded system – NetCon, which is co-developed by the Institute of Automation, Chinese Academy of Sciences, China and the Advanced Control & Network Technology Research Unit, University of Glamorgan, UK. More information about NetCon can be found at `http://system.research.glam.ac.uk`.

### 5.2.1 NetCon system

NetConController is the hardware part of the NetCon system. It consists of a main board, an AD/DA board, and a programmable IO (PIO) board. The main board has a 32-bit ARM RISC CPU (200MHz), a 64M memory, a network port, a RS-232 port, and 2 USB ports. The AD/DA board has 12 AD channels and 4 DA channels. The PIO board has 8 ports. NetConController is running on a Linux 2.4-based operating system (OS).

It is known that MATLAB/Simulink is widely used to simulate control systems. One of the Simulink components, Real-Time Workshop (RTW), makes it convenient and quick to implement the control systems in real-time by translating the Simulink

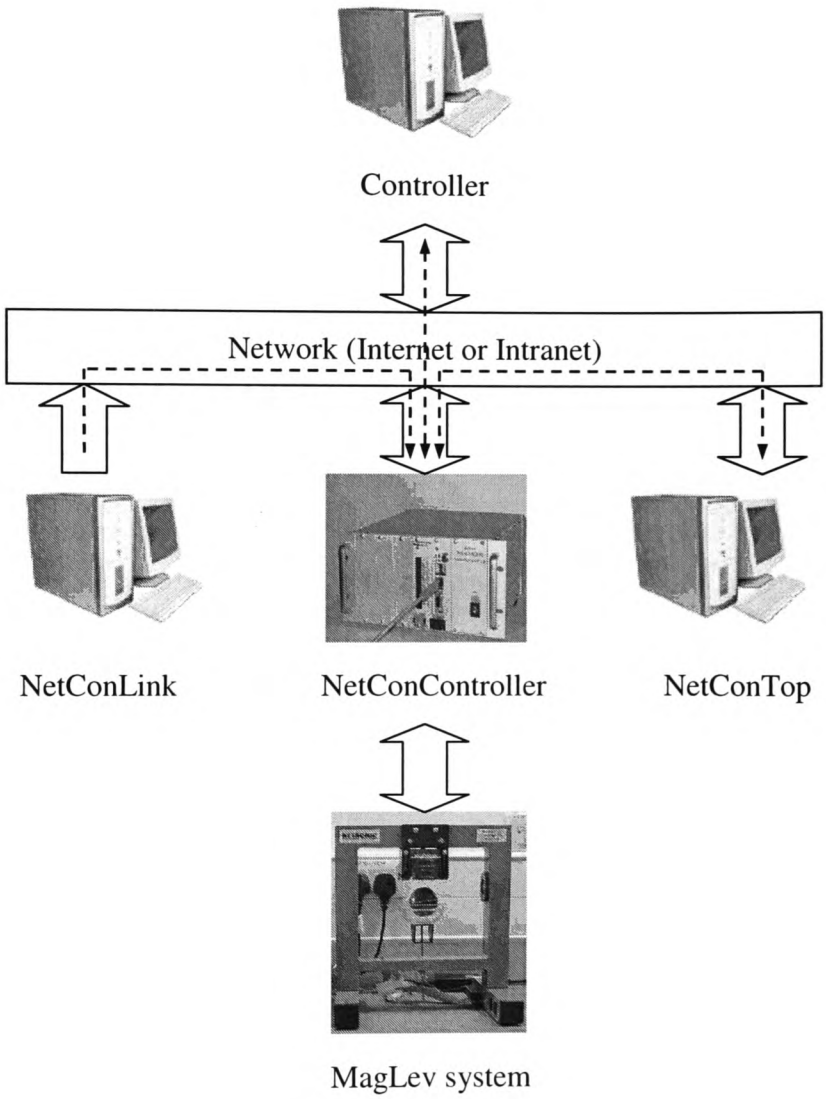



Figure 5.1: MagLev test rig

model (block diagram) into hardware-related executable codes using RTW target template files. NetConLink includes such a RTW target template file. The generated codes are automatically downloaded to the NetConController through the network. NetConLink also provides an interface for NetConTop to access parameters and signals of the running codes on NetConController.

NetConTop is a kind of SCADA (Supervisory Control And Data Acquisition) software. It provides a convenient way to build a visual diagram to monitor the operating conditions and tune the parameters of the real-time control system which is running on NetConController.

### 5.2.2 PC-based controller

The networked controller runs on a remote PC (Pentium 2.8GHz) in this test rig, which is programmed using Visual C++. The program includes not only the control law but also the communication with NetConController to receive the system feedbacks and send the control signal. It is worth pointing out that the controller can be built up in a MATLAB/Simulink environment and run on another NetConController, though the speed of the NetConController might limit the implementation of some computing-intensive control laws. Thus, all the control signals and parameters can be accessed and tuned via NetConTop.

This test rig is also easy to be transferred to the local control case by implementing the whole system in one NetConController.

In this MagLev test rig, the control system is divided into two parts by the network: the controller part and the plant part. The controller runs on a remote PC, and therefore only the plant-related Simulink block diagram is dealt with using NetConLink. This Simulink block diagram mainly includes configuration of two AD channels and one PIO port for ball position, coil current and PWM signal of MagLev system, respectively, and configuration of communication between NetConController and remote control law. The position, coil current and PWM duty cycle are monitored using NetConTop.

## 5.3 Modeling of MagLev system

The BYTRONIC MagLev system mainly consists of power interface, electro-magnet, hollow steel ball, ball position sensor and coil current sensor. Its schematic diagram is shown in Figure 5.2.

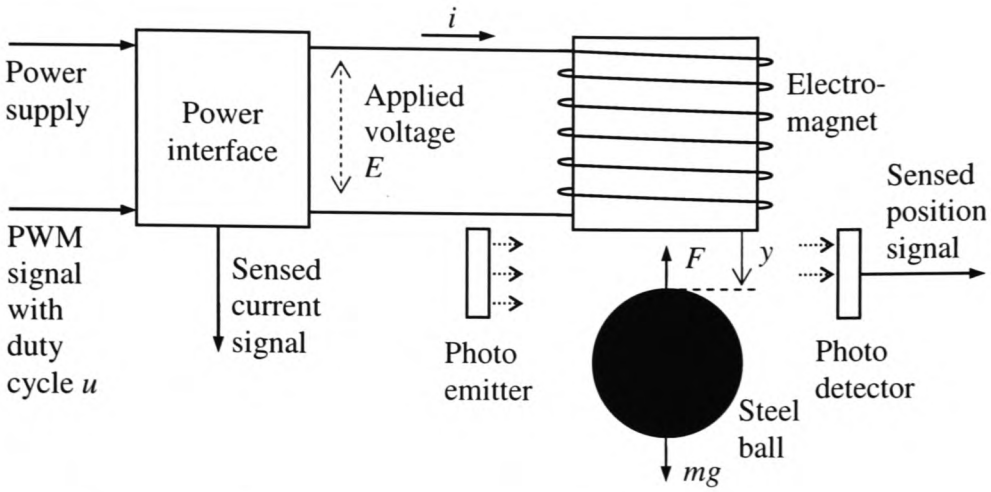



Figure 5.2: Schematic diagram of BYTRONIC MagLev system

The basic principle of this system is to apply a voltage to the electro-magnet to generate magnet force and then to keep the ball levitated. The ball position (air gap from electro-magnet to ball) is measured by a photoelectric sensor which detects the photo level received from a photo emitter. Amount of photo which the ball (in the middle of detector and emitter) allows to go through reflects its position. Under the steel ball is a tray which prevents the ball drops on the ground. The height of the tray is adjustable. In this MagLev system, a circuit board including a chip LMD18200 is used as a power interface between the supplied power and the power applied to the electro-magnet. This power transformation is governed by a PWM signal which is from, for example, the controller. The chip LMD18200 can also sense the coil current through the electro-magnet. Detailed information about LMD18200 can be found on `http://web.mit.edu/rec/datasheets/LMD18200.pdf`.

According to fundamental principles of dynamics, the MagLev system shown in Figure 5.2 satisfies the following equations (Barie and Chiasson, 1996; Hajjaji and

Ouladsine, 2001)

$$\begin{aligned} \frac{\mathrm{d}y(t)}{\mathrm{d}t} &= v(t) \\ m\frac{\mathrm{d}v(t)}{\mathrm{d}t} &= mg - F(i,y) \\ L(y)\frac{\mathrm{d}i(t)}{\mathrm{d}t} &= E(t) - Ri(t) \end{aligned} \tag{5.1}$$

where $y(t)$ is the ball position, $v(t)$ is the ball velocity, $i(t)$ is the coil current, $E(t)$ is the applied voltage, $F(t)$ is the electromagnetic force, $L(y)$ is the coil inductance, $m$ is the ball mass, $g$ is the gravity acceleration, and $R$ is the coil resistance. Note that $F(i,y)$ is nonlinear w.r.t. $i(t)$ and $y(t)$, so is $L(y)$ w.r.t. $y(t)$.

In this MagLev system, nonlinearity characteristics in $F$ and $L$ are respectively described in BYTRONIC (2006) by

$$F(i,y) = F_{em}(i,y) = i(t)^2 \frac{F_{emP1}}{F_{emP2}} e^{-y(t)/F_{emP2}} \tag{5.2}$$

$$L(y) = Rf_i(y) = R\frac{f_{iP1}}{f_{iP2}} e^{-y(t)/f_{iP2}}. \tag{5.3}$$

Furthermore, the relation of the applied voltage $E$ and the PWM duty cycle $u$ is explicitly expressed as

$$\frac{E(t)}{R} = k_i u(t) + c_i. \tag{5.4}$$

Clearly, $u$ is in the interval $[0,1]$. In equations (5.2), (5.3) and (5.4), parameters $F_{emP1}$, $F_{emP2}$, $f_{iP1}$, $f_{iP2}$, $c_i$ and $k_i$ are constants determined by the characteristics of the coil, magnetic core and the ball. These constants as well as $m$ and $g$ as shown in Table 5.1 are given in BYTRONIC (2006). Note that some of the above parameters need to be calibrated because of possible changes in the physical properties of the system, which can be done by taking some basic measurements (see Section 5.6). Replacing corresponding items in (5.1) with (5.2), (5.3) and (5.4), the system model is converted to an affine nonlinear state-space model

$$\dot{x}_1(t) = x_2(t)$$

Table 5.1: Parameters of BYTRONIC MagLev system.

| Parameters | Values | Units |
|---|---|---|
| $m$ | $5.7 \times 10^{-2}$ | [kg] |
| $g$ | 9.81 | [m/s$^2$] |
| $F_{emP1}$ | $1.75 \times 10^{-2}$ | [H] |
| $F_{emP2}$ | $5.82 \times 10^{-3}$ | [m] |
| $f_{iP1}$ | $1.41 \times 10^{-4}$ | [m·s] |
| $f_{iP2}$ | $4.56 \times 10^{-3}$ | [m] |
| $c_i$ | $2.43 \times 10^{-2}$ | [A] |
| $k_i$ | 2.52 | [A] |

$$\begin{aligned} \dot{x}_2(t) &= -\frac{1}{m}\frac{F_{emP1}}{F_{emP2}} e^{-x_1(t)/F_{emP2}} x_3(t)^2 + g \\ \dot{x}_3(t) &= \frac{f_{iP2}}{f_{iP1}} e^{x_1(t)/f_{iP2}} \left(k_i u(t) + c_i - x_3(t)\right). \end{aligned} \tag{5.5}$$

where $x = [x_1\ x_2\ x_3]^T$ is the system state with $x_1 = y$, $x_2 = v$ and $x_3 = i$ and $u$ is the input.

## 5.4 Networked feedback linearization predictive control

The design of networked control of the MagLev system is concerned with two aspects. One is the nonlinearity feature of the process, and the other is the network-induced delay. For the former, control methods based on feedback linearization model and local linearization model of are discussed (5.5). For the latter, a networked predictive control strategy is applied.

For the network-induced delay $\tau(t)$, which is defined in Chapter 4, is the round-trip delay which includes sensor-to-controller delay $\tau_{sc}(t)$ and controller-to-actuator delay $\tau_{ca}(t)$, and their discrete-time versions w.r.t. the sampling period are $\tau(k)$, $\tau_{sc}(k)$ and $\tau_{ca}(k)$ respectively. $\tau(k)$ has a upper-bound $\bar{\tau}$.

In this MagLev system, ball position and coil current are both measured. Ball velocity can be approximately obtained by numerically differentiating the position

signal at sampling instants, i.e.,

$$x_2(k) = \frac{x_1(k) - x_1(k-1)}{h}$$

where $h$ is the sampling period. Thus, the system has full state feedback.

### 5.4.1 Feedback linearization

In model (5.5), using the following nonlinear transformation coordinates reported by Chiasson (1989); Isidori (1989); Nijmeijer and Schaft (1990)

$$\begin{aligned} x_{fl1}(t) &= x_1(t) \\ x_{fl2}(t) &= x_2(t) \\ x_{fl3}(t) &= -\frac{1}{m}\frac{F_{emP1}}{F_{emP2}} e^{-x_1(t)/F_{emP2}} x_3(t)^2 + g. \end{aligned} \tag{5.6}$$

Note that the state $x_{fl3}$ stands for the ball acceleration. In this new coordinates, the system model changes to

$$\begin{aligned} \dot{x}_{fl1}(t) &= x_{fl2}(t) \\ \dot{x}_{fl2}(t) &= x_{fl3}(t) \\ \dot{x}_{fl3}(t) &= \alpha(x(t)) + \beta(x(t))u(t) \end{aligned} \tag{5.7}$$

where

$$\begin{aligned} \alpha(x(t)) &= \frac{1}{m}\frac{F_{emP1}}{F_{emP2}^2} e^{-x_1(t)/F_{emP2}} x_2(t) x_3(t)^2 \\ &\quad -\frac{2}{m}\frac{F_{emP1}}{F_{emP2}}\frac{f_{iP2}}{f_{iP1}} e^{x_1(t)/f_{iP2} - x_1(t)/F_{emP2}} x_3(t)\left(c_i - x_3(t)\right) \end{aligned} \tag{5.8}$$

$$\beta(x(t)) = -\frac{2k_i}{m}\frac{F_{emP1}}{F_{emP2}}\frac{f_{iP2}}{f_{iP1}} e^{x_1(t)/f_{iP2} - x_1(t)/F_{emP2}} x_3(t). \tag{5.9}$$

In (5.7), if $u(t)$ is designed as

$$u(t) = \frac{-\alpha(x(t)) + u_{fl}(t)}{\beta(x(t))}, \tag{5.10}$$

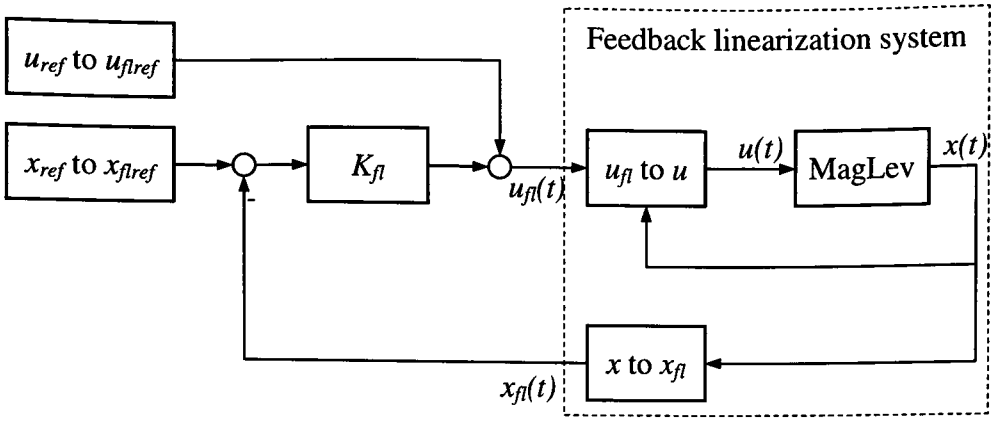



Figure 5.3: Feedback linearization control system structure without inclusion of the network.

then the resulting system becomes linear, so that

$$\dot{x}_{fl}(t) = A_{fl}x_{fl}(t) + B_{fl}u_{fl}(t) \tag{5.11}$$

with

$$A_{fl} = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{bmatrix}, B_{fl} = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}.$$

Note that (5.10) is feasible because $x_3(t)$ (coil current $i(t)$) is greater than 0. Clearly, feedback linearization model (5.11) is controllable, so that

$$u_{fl}(t) = K_{fl}\left(x_{flref} - x_{fl}(t)\right) + u_{flref}. \tag{5.12}$$

When considering local control (without network), static state-feedback control $K_{fl}$ can be designed using many existing methods, such as pole-placement, optimization. In (5.12), $x_{flref}$ and $u_{flref}$ are references which are corresponding to $x_{ref}$ and $u_{ref}$ and can be obtained from (5.6) and (5.8), (5.9) and (5.10), respectively.

Figure 5.3 shows the system control structure after the above feedback linearization.

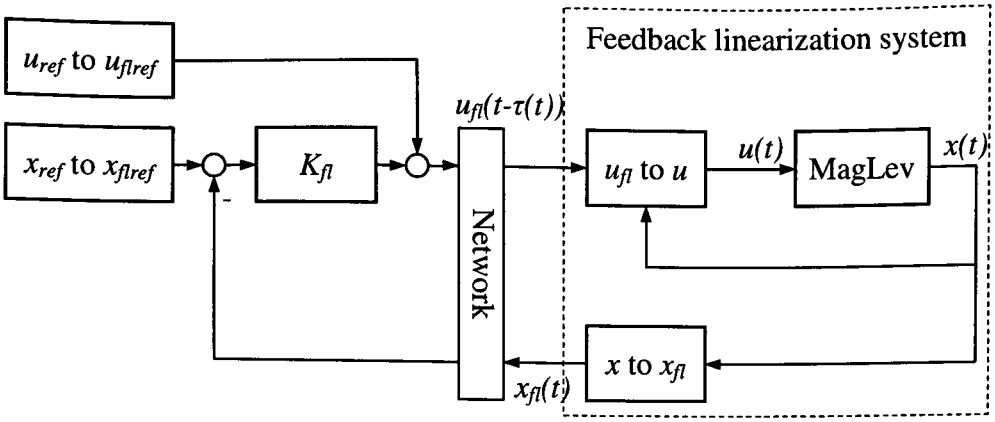



Figure 5.4: Feedback linearization control system structure with network.

### 5.4.2 Networked feedback linearization predictive control

When considering networked control, where the network channel is placed in Figure 5.3 leads to quite different results. If the network is put after $x(t)$ and before $u(t)$, Clearly the delayed control signal is

$$u(t-\tau(t)) = \frac{-\alpha(x(t-\tau(t))) + u_{fl}(t-\tau(t))}{\beta(x(t-\tau(t)))}.$$

And therefore equation $\dot{x}_{fl3}(t)$ in (5.7) cannot be linearized. That is, the system is no longer feedback linearizable. If the network is placed before $u_{fl}(t)$ and after $x_{fl}(t)$, as shown in Figure 5.4, the delayed control signal $u_{fl}(t-\tau(t))$ still linearizes (5.7) but introduces an input delay, i.e.,

$$\dot{x}_{fl}(t) = A_{fl}x_{fl}(t) + B_{fl}u_{fl}(t-\tau(t)). \tag{5.13}$$

In this structure, one disadvantage is that the two blocks "$u_{fl}$ to $u$" and "$x$ to $x_{fl}$" require some calculation on the plant side. For (5.13), control gain $K_{fl}$ has to tolerate $\tau(t)$ as large as possible. Many literature addresses this problem in NCS, for example Zhang et al. (2001); Yue et al. (2004); He et al. (2006). Here, in order to actively compensate for the network-induced delay, networked predictive method (referring to Section 4.6.2) is adopted. Firstly, sampling (5.11) with sampling period $h$ gives

$$x_{fl}(k+1) = \Phi_{fl}x_{fl}(k) + \Gamma_{fl}u_{fl}(k), \tag{5.14}$$

where $\Phi_{fl} = e^{A_{fl}h}$ and $\Gamma_{fl} = \int_0^h e^{A_{fl}\eta}d\eta B_{fl}$. Then, at the controller side, denote the received state as $z(k)$, which is the delay version of $x_{fl}(k)$, i.e., $z(k) = x_{fl}(k-\tau_{sc}(k))$. Then, according to a feedback gain $K_{fl}$ in (5.18) and system model $\Phi$ and $\Gamma$ in (5.16), the control prediction sequence up to the maximum round-trip delay $\bar{\tau}$ is

$$V(k) = \begin{bmatrix} v(k|k) \\ v(k+1|k) \\ \vdots \\ v(k+\bar{\tau}|k) \end{bmatrix}$$

which is obtained by recursively calculating the following two equations (Hu et al., 2007)

$$\begin{cases} z(k+i+1|k) = \Phi_{fl}z(k+i|k) + \Gamma_{fl}v(k+i|k) & i = 0, 1, \cdots, \bar{\tau}-1 \\ v(k+i|k) = K_{fl}\left(x_{flref} - z(k+i|k)\right) + u_{flref} & i = 0, 1, \cdots, \bar{\tau} \end{cases}. \tag{5.15}$$

The whole prediction sequence $V(k)$ is sent back to MagLev system as

$$V(k-\tau_{ca}(k)) = \begin{bmatrix} v(k-\tau(k)|k-\tau(k)) \\ v(k-\tau(k)+1|k-\tau(k)) \\ \vdots \\ v(k-\bar{\tau}|k-\tau(k)) \end{bmatrix}.$$

According to the actual measured round-trip delay $\tau(k)$, the proper element $v(k|k-\tau(k))$ is chosen as the control input, i.e., $u(t) = v(k|k-\tau(k))$. The design of $K_{fl}$ is discussed in Chapter 4.

## 5.5 Networked direct linearization predictive control

### 5.5.1 Direct linearization

Given an operating point $x_0 = [x_{10}, x_{20}, x_{30}]^T$ and $u_0$, model (5.5) is linearized as

$$\dot{x}(t) = A_{dl}x(t) + B_{dl}u(t) \tag{5.16}$$

with

$$A_{dl} = \begin{bmatrix} 0 & 1 & 0 \\ a_{21} & 0 & a_{23} \\ a_{31} & 0 & a_{33} \end{bmatrix}, B_{dl} = \begin{bmatrix} 0 \\ 0 \\ b_3 \end{bmatrix}$$

$$a_{21} = \frac{x_{30}^2}{m}\frac{F_{emP1}}{F_{emP2}^2}e^{-\frac{x_{10}}{F_{emP2}}}$$

$$a_{23} = -\frac{2x_{30}}{m}\frac{F_{emP1}}{F_{emP2}}e^{-\frac{x_{10}}{F_{emP2}}}$$

$$a_{31} = -(k_i u_0 + c_i - x_{30})\left(\frac{f_{iP1}}{f_{iP2}^2}e^{-\frac{x_{10}}{f_{iP2}}}\right)^2$$

$$a_{33} = -\frac{f_{iP2}}{f_{iP1}}e^{\frac{x_{10}}{f_{iP2}}}$$

$$b_3 = k_i\frac{f_{iP2}}{f_{iP1}}e^{\frac{x_{10}}{f_{iP2}}}.$$

Correspondingly, the discrete-time form of (5.16) with sampling period $h$ is

$$x(k+1) = \Phi_{dl}x(k) + \Gamma_{dl}u(k), \tag{5.17}$$

where $\Phi_{dl} = e^{A_{dl}h}$ and $\Gamma = \int_0^h e^{A_{dl}\eta}d\eta B_{dl}$.

Similarly, using a state-feedback control law results in

$$u(k) = K_{dl}\left(x_{ref} - x(k)\right) + u_{ref}. \tag{5.18}$$

Table 5.2: Relation between ball position and sensor output.

| ball position (mm) | 0 | 0.7 | 1.4 | 2.1 | 2.8 | 3.5 | 4.2 | 4.9 |
|---|---|---|---|---|---|---|---|---|
| sensor output (V) | 9.13 | 9.12 | 9.12 | 9.12 | 9.12 | 9.11 | 9.10 | 9.10 |
| ball position (mm) | 5.6 | 6.3 | 7 | 7.7 | 8.4 | 9.1 | 9.8 | 10.5 |
| sensor output (V) | 9.10 | 9.09 | 9.08 | 9.03 | 8.93 | 8.76 | 8.52 | 8.19 |
| ball position (mm) | 11.2 | 11.9 | 12.6 | 13.3 | 14 | 14.7 | 15.4 | 16 |
| sensor output (V) | 7.75 | 7.2 | 6.55 | 5.84 | 5.15 | 4.63 | 4.29 | 4.11 |
| ball position (mm) | 16.8 | 17.5 | 18.2 | 18.9 | 19.6 | 20.3 | 21 | 21.7 |
| sensor output (V) | 4 | 3.93 | 3.87 | 3.83 | 3.81 | 3.8 | 3.8 | 3.79 |

### 5.5.2 Networked direct linearization predictive control

As for the networked predictive control version, recursive calculation of the control prediction sequence (5.15) is correspondingly modified as

$$\begin{cases} z(k+i+1|k) = \Phi_{dl} z(k+i|k) + \Gamma_{dl} v(k+i|k), & i = 0, 1, \cdots, \bar{\tau} - 1 \\ v(k+i|k) = K_{dl} \left( x_{flref} - z(k+i|k) \right) + u_{flref}, & i = 0, 1, \cdots, \bar{\tau}. \end{cases}$$

## 5.6 Simulation and experiments

### 5.6.1 Calibration of MagLev system

The output of the ball position sensor and coil current sensor needs to be calibrated to the actual position and current.

The calibration of the position sensor is done as follows. Firstly, fix the ball on a screw. By turning the screw (one turning of screw equivalent to 0.7mm position variation), a series of ball position and sensor output are measured. Table 5.2 gives this data. The position value corresponding to the sensor output which is not in the table is calculated by linear interpolation. The relationship is demonstrated in Figure 5.5. It can be seen that a good operating point is between 8 $\sim$ 15mm, because in this range the resolution is the best.

Similar to the position sensor, Table 5.3 gives the relationship between PWM duty cycle, coil current and its sensor output. During the calibration, the constraints of position, current and PWM duty cycle are also determined.

Note that the MagLev system is subject to some physical constraints. The control

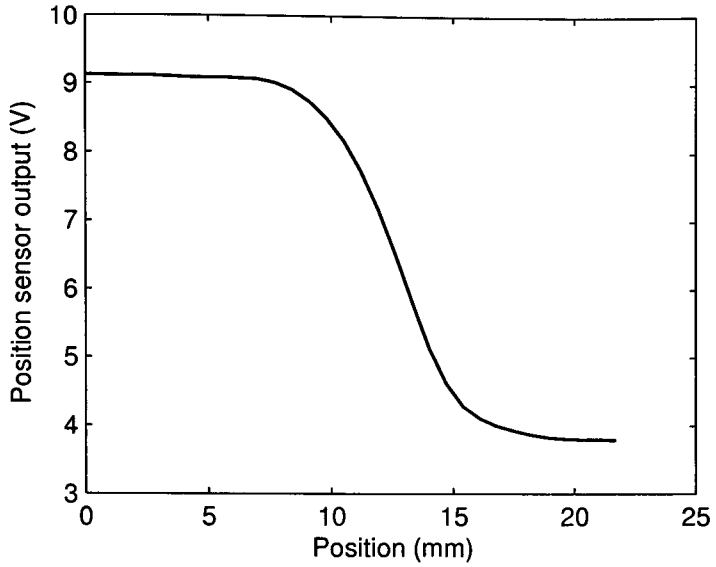



Figure 5.5: Relation between ball position and sensor output.

Table 5.3: Relation among duty cycle, coil current and its sensor output.

| PWM duty cycle (%) | 100 | 90 | 80 | 70 | 60 | 50 |
|---|---|---|---|---|---|---|
| coil current (A) | 2.16 | 1.95 | 1.74 | 1.53 | 1.32 | 1.11 |
| sensor output (V) | 0.99 | 1.71 | 1.62 | 1.43 | 1.24 | 1.04 |
| PWM duty cycle (%) | 40 | 30 | 20 | 10 | 0 | |
| coil current (A) | 0.89 | 0.67 | 0.45 | 0.22 | 0.04 | |
| sensor output (V) | 0.84 | 0.63 | 0.43 | 0.22 | 0.04 | |

signal $u$ (PWM duty cycle) has a maximum value $u_{\max} = 1$. It can be seen from the third equation in (5.1) that a minimal $u_{\min}$ is required to generate an effective current. By measurement, $u_{\min} = 0.02$ is determined.

### 5.6.2 Test-rig configuration

In this chapter, the system is set up under an Intranet network. NetConLink, NetConTop and Remote controller are all run on a PC with an IP address 192.168.0.12. NetConController connected to MagLev is with an address 192.168.0.7. By measuring, the round trip network transmission delay has a maximal 0.02s and there is no data dropout happening in the network. The sampling period $h = 0.01$s is taken, so the round-trip network-induced delay has upper bound $\bar{\tau} = 2$. The nominal equilibrium point of the ball position is set to 0.01m, and according to (5.5) the nominal coil current 1.2A and duty cycle 0.4 are obtained. That is to say, $x_{ref} = [0.01\ 0\ 1.2]^T$ and $u_{ref} = 0.4$. In order to reduce violent oscillation of the ball before it tends stable, the height of the tray under the ball is set to 0.016m.

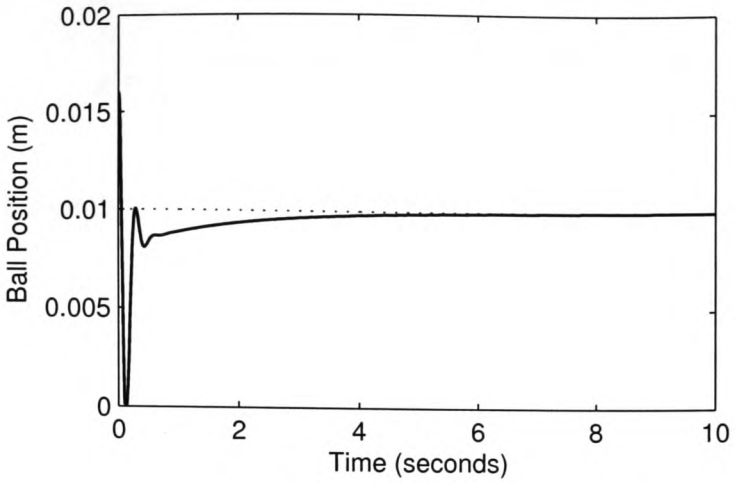
### 5.6.3 Simulation

For the feedback linearization method, controller $K_{fl} = [\ 281.77\quad 498.15\quad 21.49\ ]$ is designed for the network-free system (5.14) with

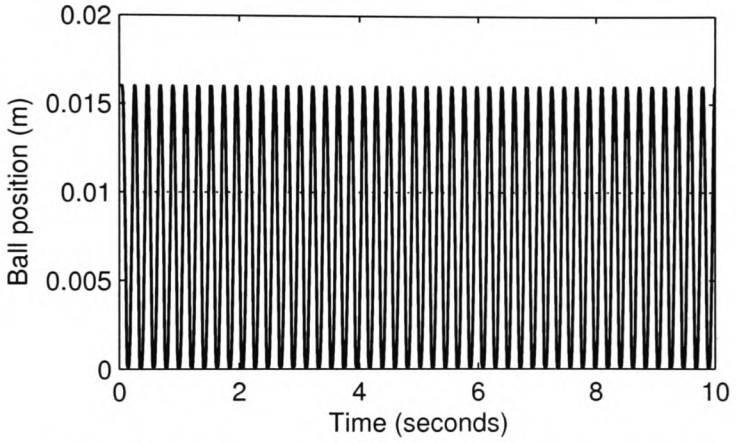
$$\Phi_{fl} = \begin{bmatrix} 1 & 0.1 & 0.0001 \\ 0 & 1 & 0.01 \\ 0 & 0 & 0 \end{bmatrix}, \Gamma_{fl} = \begin{bmatrix} 0 \\ 0.0001 \\ 0.01 \end{bmatrix}.$$

The system responses for step signal $0.01 \times 1(t)$ are shown in Figure 5.6, where (a) is for network-free case (local control), (b) is networked control without compensation, and (c) is for networked predictive control. The dash line and solid line represent the reference and the system response, respectively. It can be seen without network-induced delay compensation strategy, $K_{fl}$ can not make the system stable.

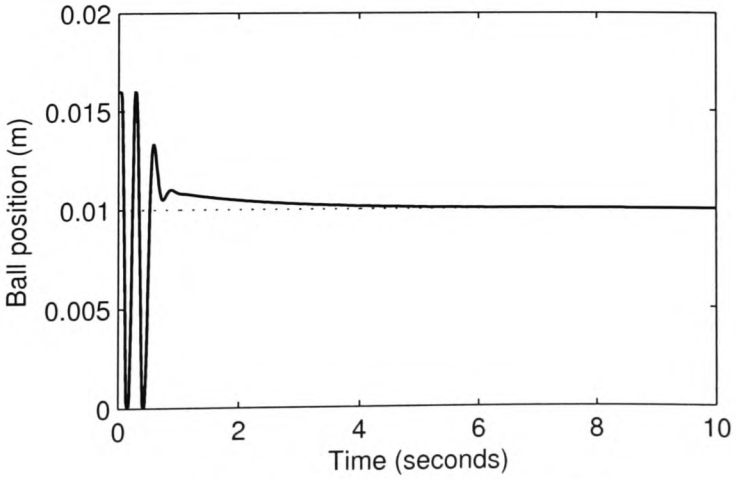
For the direct linearization method, at nominal point $x_{ref}$ and $u_{ref}$, the linearization model is obtained as




(a) local control




(b) networked control without delay compensation




(c) networked predictive control

Figure 5.6: Simulation of step signal $0.01 \times 1(t)$ tracking of ball position with feedback linearization control.

$$A_{dl} = \begin{bmatrix} 0 & 1 & 0 \\ 1684.7 & 0 & -19.3 \\ 0 & 0 & -288.8 \end{bmatrix}, B_{dl} = \begin{bmatrix} 0 \\ 0 \\ 726.7 \end{bmatrix}.$$

Further, with sampling period $h = 0.01$, its discrete model is given by

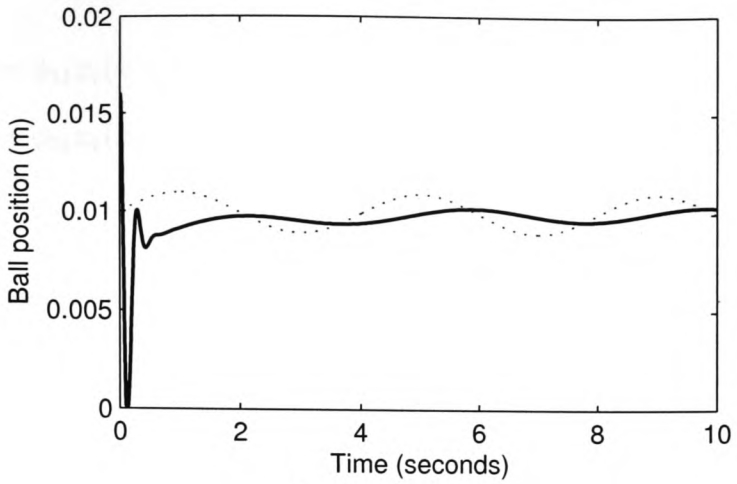
$$\Phi_{dl} = \begin{bmatrix} 1.0854 & 0.0103 & -0.0005 \\ 17.3239 & 1.0854 & -0.066 \\ 0 & 0 & 0.0557 \end{bmatrix}, \Gamma_{dl} = \begin{bmatrix} -0.0013 \\ -0.3324 \\ 2.3763 \end{bmatrix}.$$

The control gain for network-free system is designed as $K_{dl} = [\ -95.6 \quad -0.53 \quad 0.238\ ]$. For a sine signal $0.01 + 0.001 \times \sin(\frac{\pi}{2}t)$, the system responses are shown in Figure 5.7, where (a), (b) and (c) are for local control, networked control without delay-compensation and networked predictive control. Again, the NPC method is able to compensate the delay and make the system stable.

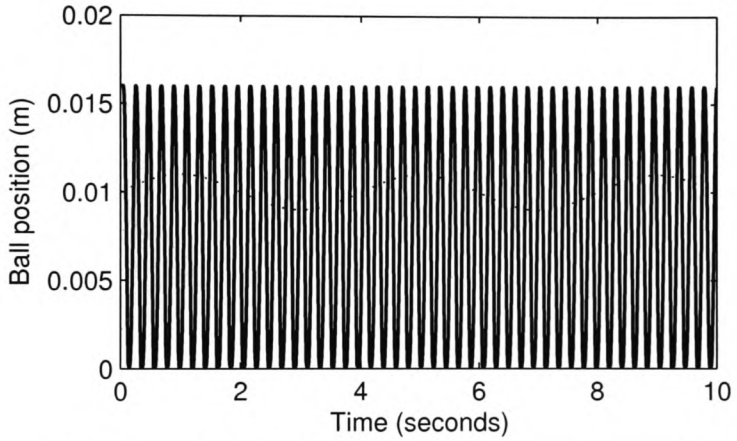
### 5.6.4 Experiments

In the above methods, the control prediction sequence is very dependent on model accuracy. When the above two predictive control methods are implemented in the real-time system, it is found that neither of them can make the practical system stable. This is because the model is not built very accurately and the MagLev system has the time-varying nature. Thus the result obtain from (5.16) is not accurate enough to be applied to NPC. It is reasonable to apply on-line model parameter tuning to improve the model accuracy. The feedback linearization method uses the nonlinearization model which is very difficult to tune on-line, so here only the direct linearization method is discussed.

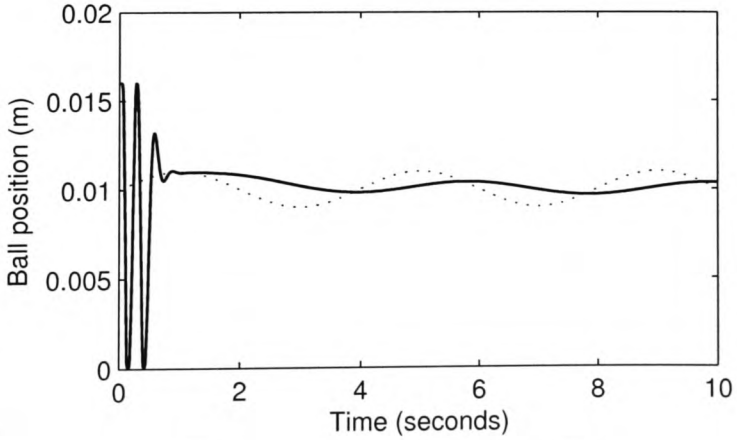
The recursive least square (RLS) method is often used for online parameter estimation and they are adopted here. Expand the state-space model (5.17) to




(a) local control




(b) networked control without delay compensation




(c) networked predictive control

Figure 5.7: Simulation of sine signal $0.01 + 0.001 \sin(\frac{\pi}{2}t)$ tracking of ball position with direct linearization control.

three separated equations

$$\begin{cases} x_1(k) = \phi_{11}x_1(k-1) + \phi_{12}x_2(k-1) + \phi_{13}x_3(k-1) + \gamma_1 u(k-1) \\ x_2(k) = \phi_{21}x_1(k-1) + \phi_{22}x_2(k-1) + \phi_{23}x_3(k-1) + \gamma_2 u(k-1) \\ x_3(k) = \phi_{31}x_1(k-1) + \phi_{32}x_2(k-1) + \phi_{33}x_3(k-1) + \gamma_3 u(k-1) \end{cases} . \quad (5.19)$$

For the first equation, it can be rewritten as

$$x_1(k) = \varphi^T(k)\theta,$$

where $\varphi(k) = [x_1(k-1), x_2(k-1), x_3(k-1), u(k-1)]^T$ are past states and control and $\theta = [\phi_{11}, \phi_{12}, \phi_{13}, \gamma_1]^T$ is the parameter vector to be estimated. Thus, the RLS algorithm is obtained as

$$\begin{cases} \hat{\theta}(k) = \hat{\theta}(k-1) + L(k)\hat{\varepsilon}(k) \\ \hat{\varepsilon}(k) = x_1(k) - \varphi^T(k)\hat{\theta}(k-1) \\ L(k) = \dfrac{P(k-1)\varphi(k)}{\lambda + \varphi^T(k)P(k-1)\varphi(k)} \dfrac{P(k-1)\varphi(k)}{\lambda+\varphi^T(k)P(k-1)\varphi(k)} \\ P(k) = \left(P(k-1) - \dfrac{P(k-1)\varphi(k)\varphi^T(k)P(k-1)}{\lambda(k) + \varphi^T(k)P(k-1)\varphi(k)}\right)/\lambda \end{cases}$$

where $\lambda$ $(0 < \lambda < 1)$ is called the forgetting factor. Initial $\theta(0)$ can take the value from (5.17) and $P(0) = \rho I$ with a large $\rho$. Other parameters in $\Phi_{dl}$ and $\Gamma_{dl}$ can be estimated by a similar procedure using the second and third equations in (5.19).

In (5.15), $\Phi_{dl}$ and $\Gamma_{dl}$ should take the estimated values when online parameter estimation is applied. In this case, it can be seen from (5.19) that not only $x(k)$ but also $x(k-1)$ and $u(k-1)$ should be sent to the remote controller from the MagLev system. Note that because of the network-induced delay, the estimated model $\hat{\Phi}_{dl}(k - \tau_{sc}(k))$ and $\hat{\Gamma}_{dl}(k - \tau_{sc}(k))$ does not reflect the actual system in time, but generally it at least provides an "improved" model. On the other hand, it might be argued that $x(k-1)$ is not necessarily sent with $x(k)$ and $u(k-1)$ because it has already been sent out at $k-1$. This is not true, because the randomness of network-transmission delay or data dropout can not make sure that $x(k - \tau_{sc}(k))$

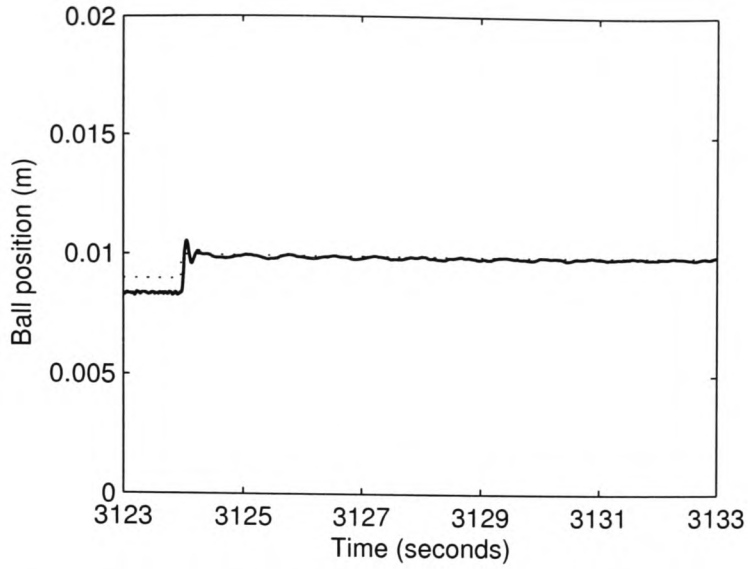



Figure 5.8: Experiments of setpoint response of ball position 0.01m using adaptive NPC based on direct linearization model

and $x(k-\tau_{sc}(k)-1)$ is always available at the same time, which stops the processing of the online estimation algorithm.

Take the $\Phi_{dl}$ and $\Gamma_{dl}$ calculated in Section 5.6.3 as the initial values to tune the model. Control results using adaptive networked directive linearization predictive method for setpoint response and sine signal tracking of the ball position are shown in Figure 5.8 and Figure 5.9 respectively, where the forgetting factor $\lambda$ takes 0.98 and $\rho$ takes 10000. Figure 5.8 is the setpoint response for step signal $0.009 + 0.001 \cdot 1(t)$ and Figure 5.9 is the tracking for the sine signal $0.01 + 0.001 \times \sin(\frac{\pi}{2})$. The dash line and solid line represent the reference and the system response, respectively. These results show the NPC method with on-line model parameter modification is applicable for the real-time implementation of the MagLev system.

## 5.7 Summary

In this chapter, a magnetic levitation system test rig has been reported. It can be used as a platform for testing control algorithms for nonlinear, time-varying and fast system over the network or just using local control. This chapter has introduced several control methods and implemented them on this test rig. The simulation and

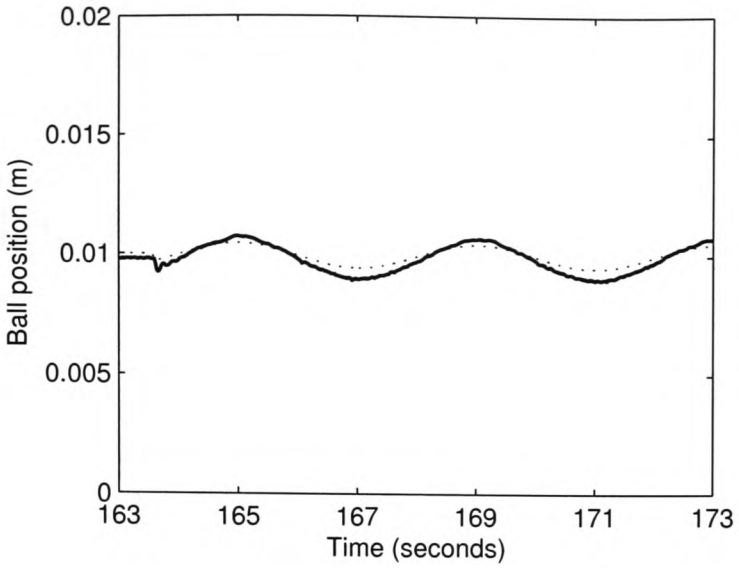



Figure 5.9: Experiment of sine signal $0.01 + 0.001\sin(\frac{\pi}{2}t)$ tracking of ball position using adaptive NPC based on direct linearization model

experiment results show that the networked predictive method is effective for NCSs.

# Chapter 6

# Implementation of Networked Control Systems Using NCS Toolbox

This chapter introduces a MATLAB/Simulink toolbox for networked control systems. It shows how the toolbox is designed to incorporate control schemes, network simulation, network interface, and plant interface. This toolbox can be applied to not only the simulation study but also the real-time application of NCSs. Various control schemes are discussed and compared within the toolbox framework. Using the proposed NCS toolbox, the implementation of NCSs can be fulfilled very quickly so that users can focus on the study of control schemes.

## 6.1 Introduction

Networked control systems (NCSs), wherein the actuator and the sensor are connected to the controller via a communication network, have attracted much attention from both theoretical studies and practical applications in recent years (Zhang et al., 2001; Tipsuwan and Chow, 2003; Yang, 2006). NCSs provide a natural framework to realize distributed control, remote control and resource sharing, etc, with low cost. Compared to traditional control systems, however, in NCSs, the involvement of the network in the control loop brings some extra issues, such as how to describe the network transmission mechanism and then how to deal with the network-transmission delay, data disorder and dropout, clock asynchronism, and so on (Montestruque and Antsaklis, 2004; Yang et al., 2003; Hespanha, 2006; He et al., 2007). These issues not only make the control design more complicated, but also make the system implementation more challenging. Current research activities in NCSs can be divided into three categories: control system analysis and control scheme design; network architecture, protocol and scheduling; simulation and real-time studies.

An appropriate control scheme is vital for NCSs to achieve a desired control performance, which depends on a good understanding of all issues induced by the network. In order to verify the effectiveness of a designed control scheme, simulation and real-time study is necessary. NS2 (network simulation 2), `http://www.isi.edu/nsnam/ns/index.html`, is a discrete event simulator targeted at networking research. It provides substantial support for the simulation of TCP, routing, and multicast protocols over wired or wireless and local or Internet networks. However, NS2 does not support MATLAB, which limits its application in control-oriented research. TrueTime (Cervin et al., 2003) is a MATLAB/Simulink based toolbox, which consists of a kernel block which simulates real-time environment I/O converters, network interfaces and event interrupts, and a network block where many parameters about the internal network transmission mechanism can be configured. However, this network block emphasizes the simulation of the internal network features which are not entirely appropriate or straightforward for the design of control schemes. NCS-sim, `http://www.sussex.ac.uk/Users/taiyang/`, is another toolbox by which users can visually configure the external network fea-

tures, such as the network transmission delay, data dropout, etc. Both TrueTime and NCS-sim, however, are not applicable to real-time implementation. Some software packages devoted to this subject have been considered by a number of authors. For example, Teng (2002) compared different realization of real-time control based on MATLAB/Simulink. Kim et al. (2006b) built up a server-client structure for real-time NCS, which is not based on MATLAB though.

Based on the above publications, this chapter considers a MATLAB/Simulink based NCS toolbox which provides a framework for both the simulation and real-time implementation of NCSs. Firstly, a general NCS structure is studied in detail. It is correspondingly divided into several parts according to their functions. Then these functions are realized using MATLAB/Simulink blocks as a toolbox. Thus, implementing a NCS becomes as easy as implementing a traditional control system. Moreover, with this toolbox, only little change is needed when transfer the simulation implementation to the real-time implementation.

## 6.2 Simulation and real-time implementation of NCSs

In a traditional control system, shown in Figure 6.1, the controller and the plant (actuator and sensor) exchange information directly. In NCSs, however, a communication network is placed in the middle of information exchanging channels, which is shown in Figure 6.2. In Chapter 4, the impact of the network on the closed-loop system is discussed in term of the network-induced delay, which presents challenges for the design of control schemes. From the perspective of the implementation of a NCS, the network causes the following issues:

- A simulation of the network is needed to represent the data flow on the network when doing NCS simulation study;
- In addition, two extra units, data receiving and data sending, are also required to deal with the data exchange with the network;
- A data processing unit is necessary in the whole control scheme to cope with the network-transmission delay, data dropout or data disorder so that it is

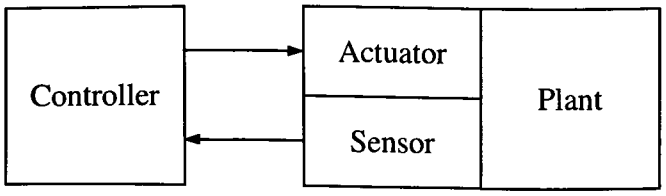



Figure 6.1: Traditional control system structure.

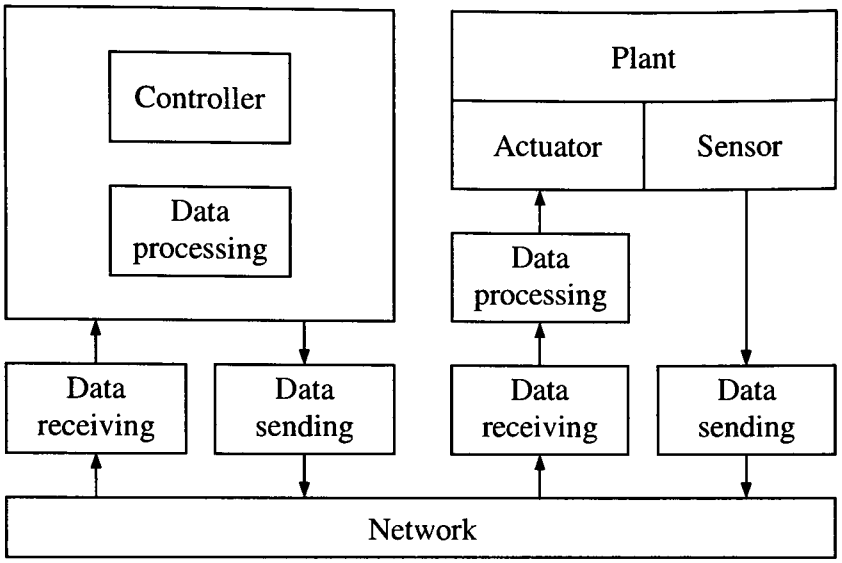



Figure 6.2: Typical NCS structure.

able to provide an appropriate input to the controller or to the actuator;

- Further, the controller design has to consider the impact of network-transmission delay and data dropout.

### 6.2.1 Network simulation

A communication network can be described from two perspectives. One is by the internal network features including network architecture, bandwidth, protocol, scheduling, and so on. The other one is by the external features mainly about the network-transmission delay and data dropout. The external features are the result of the internal features. The online adjustment of the network protocol and scheduling according to different network situation can alleviate the transmission delay and data dropout somewhat and thus improve the system performance. This belongs to the category of co-design of networks and control. Internal network features de-

scribe the network better but require considerable knowledge of the network working principles. Whereas, external features are easy to obtain by doing some simple experiments on the network. This is similar to modeling a system using state-space method or input-output method.

From the control point of view, network external features are of more concern than its internal features. Based on this, the network is simulated as follows. When data is sent to the network, a test is firstly taken to decide whether data is dropped. The occurrence of dropping is according to a specified data dropout rate. Then for undropped-out data, it is allocated a network-transmission delay. This delay indicates how long the data will wait before it is propagated out of the network. Network-transmission delay is usually random but with lower and upper bounds. After the delay is allocated, the data with its propagating-out time which is the sum of the current time and the allocated delay is put in a queuing buffer. In the end, the propagating-out time of each data in the queuing buffer is compared to the current time. Those data whose propagating-out time equal to the the current time are sent out of the network. Clearly, the buffer size is equal to the upper bound of the network-transmission delay. Figure 6.3 summarizes the above procedures. It can be seen that two features of the network, i.e., transmission delay and data-dropout rate, are enough to simulate the network.

In Figure 6.3, define a function

$$rand()$$

which generates a uniform distribute random number in the range $[0, 1]$. Give the data dropout rate $\rho$ $(0 \leq \rho < 1)$, and then when

$$rand() \leq \rho,$$

the corresponding data is dropped out. Similarly, assume the network-transmission delay have lower bound $\sigma_l$ and upper bound $\sigma_u$ $(\sigma_u > \sigma_l)$. Use

$$rand() \times (\sigma_u - \sigma_l) + \sigma_l$$

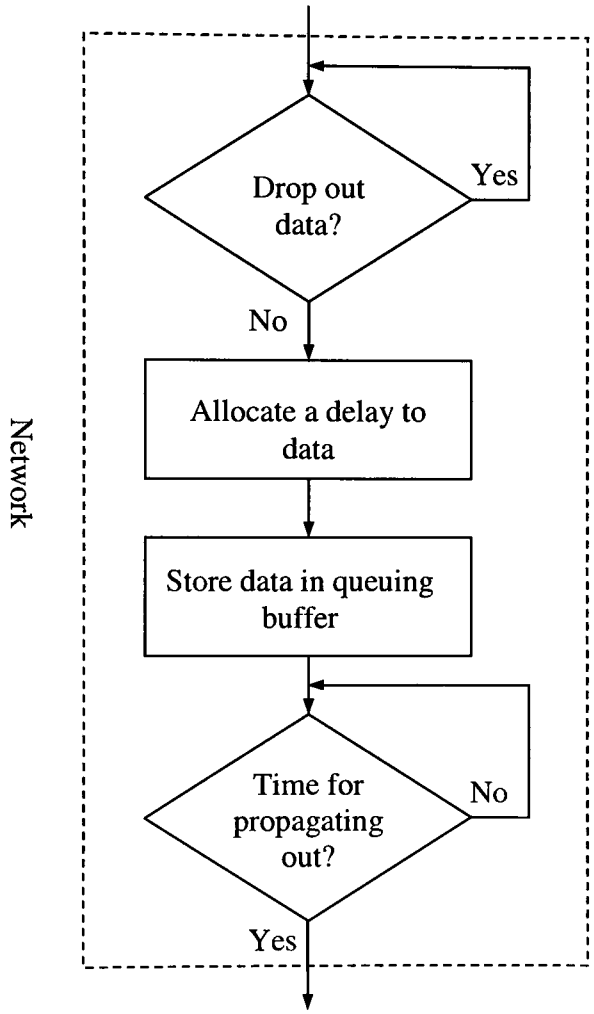



Figure 6.3: Simulation of network.

to get a random number between $\sigma_l$ and $\sigma_u$ as the random delay allocated to the data. Sometimes the random delay is restricted to have the property of Markov-chain, where the current delay depends on the previous delay and cannot increase more than one step. For example, let the previous delay and the current delay be $i, j$ $(\sigma_l \leq i, j \leq \sigma_u)$ respectively. Then the current delay $j$ satisfies $\sigma_l \leq j \leq i+1$. Moreover, the change from $i$ to $j$ is subject to a probability $p_{i,j}$. Consider all these probabilities to construct the following $(\sigma_u - \sigma_l + 1) \times (\sigma_u - \sigma_l + 1)$ matrix $\{p_{i,j}\}$

$$\begin{bmatrix} p_{\sigma_l,\sigma_l} & p_{\sigma_l,\sigma_l+1} & 0 & \cdots & 0 \\ p_{\sigma_l+1,\sigma_l} & p_{\sigma_l+1,\sigma_l+1} & p_{\sigma_l+1,\sigma_l+2} & \cdots & 0 \\ p_{\sigma_l+2,\sigma_l} & p_{\sigma_l+2,\sigma_l+1} & p_{\sigma_l+2,\sigma_l+2} & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & 0 \\ p_{\sigma_u,\sigma_l} & p_{\sigma_u,\sigma_l+1} & p_{\sigma_u,\sigma_l+2} & \cdots & p_{\sigma_u,\sigma_u} \end{bmatrix}.$$

This is called Markov-chain matrix which can be obtained by measuring the network many times and should be given in advance. In each row of the Markov-chain matrix, the probabilities satisfy

$$p_{i,j} = 0, \ i+1 < j \leq \sigma_u$$

$$p_{i,\sigma_l} + p_{i,\sigma_l+1} + \cdots + p_{i,i} + p_{i,i+1} = 1.$$

With the previous delay $i$ and matrix $\{p_{i,j}\}$, the current delay $j$ is determined as follows:

$$j = \begin{cases} \sigma_l & 0 \leq rand() \leq p_{i,\sigma_l} \\ \sigma_l + 1 & p_{i,\sigma_l} < rand() \leq p_{i,\sigma_l} + p_{i,\sigma_l+1} \\ \vdots & \\ i & p_{i,\sigma_l} + p_{i,\sigma_l+1} + \cdots + p_{i,i-1} < rand() \leq p_{i,\sigma_l} + p_{i,\sigma_l+1} + \\ & \cdots + p_{i,i-1} + p_{i,i} \\ i+1 & p_{i,\sigma_l} + p_{i,\sigma_l+1} + \cdots + p_{i,i} < rand() \leq 1 \end{cases}$$

### 6.2.2 Network interface

In Figure 6.2, the data-receiving unit and data-sending unit are composed of the network interface, which has the following basic functions:

- decoding or encoding the network data-packet;
- configuring target/source IP address and Socket Port number for data transmission;
- storing received data into a receiving buffer (only for the data-receiving unit);

From the control perspective, the first function above is not needed. The second function is not used when considering a simulation study. The mechanism of data-sending unit is relatively simple. It just transfers data in the form of network packet to the network in time order. It is worth noting that the time-stamp information is usually attached to the data before it is passed to the data-sending unit. See Figure 6.4. In the situation without causing confusion, the integrated data (data and time-stamp) is still called data. When data is propagated to its destination (target), the data-receiving unit firstly stores it in a receiving buffer. This buffer has maximum size equal to $\sigma_u - \sigma_l + 1$, which can handle the case where $\sigma_u - \sigma_l + 1$ data are propagated out of the network at the same time, these data have the network-transmission delay $\sigma_u, \sigma_u - 1, \cdots, \sigma_l$, respectively. Generally, due to the randomness of the network-transmission delay and data dropout, the number of data in the receiving buffer has the following three cases:

- one data stored;
- more than one data stored; or
- no data stored.

The contents in the receiving buffer are updated in two ways. One is that the buffer is emptied as long as it is accessed and read out by its next unit (data-processing unit; see Section 6.2.3). The other way is if the buffer is full the new coming data will expel the earliest stored data (noting it is not the oldest data concerning the time-stamp of the data). Figure 6.5 shows the procedure of data-receiving unit.

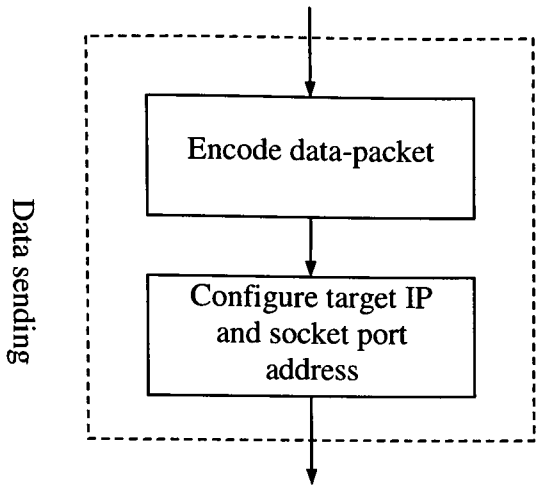



Figure 6.4: Data sending.

### 6.2.3 Control schemes

The design of a control scheme in NCSs includes not only the controller but also the data processing, i.e., the strategy of dealing with the network-induced issues. Here the data processing is of more concern. The controller design has been addressed in Chapter 4. Note that in some complicated control schemes, one part of the controller, for example the network-induced delay compensator in NPC scheme (referring to Figure 4.3), might be placed at the plant side.

As discussed in Section 6.2.2 about data propagating from data-receiving unit, the data processing unit, responsible for providing data for the controller and the actuator, encounters the following cases.

**Case1:** there is data obtained from data-receiving unit at the current time instant. The obtained data (one or more than one) is firstly compared to the data which was previously used in the controller or the actuator to determine whether the current received data is the latest data (by comparing the attached time-stamps). If the current data is latest, it is reasonable to use it in the control law or in the actuator, or otherwise,

**Case2:** if the current data is older than the previous one, the data-processing unit adopts one of the following strategies:

- to continue providing the previous data (Strategy 1); or

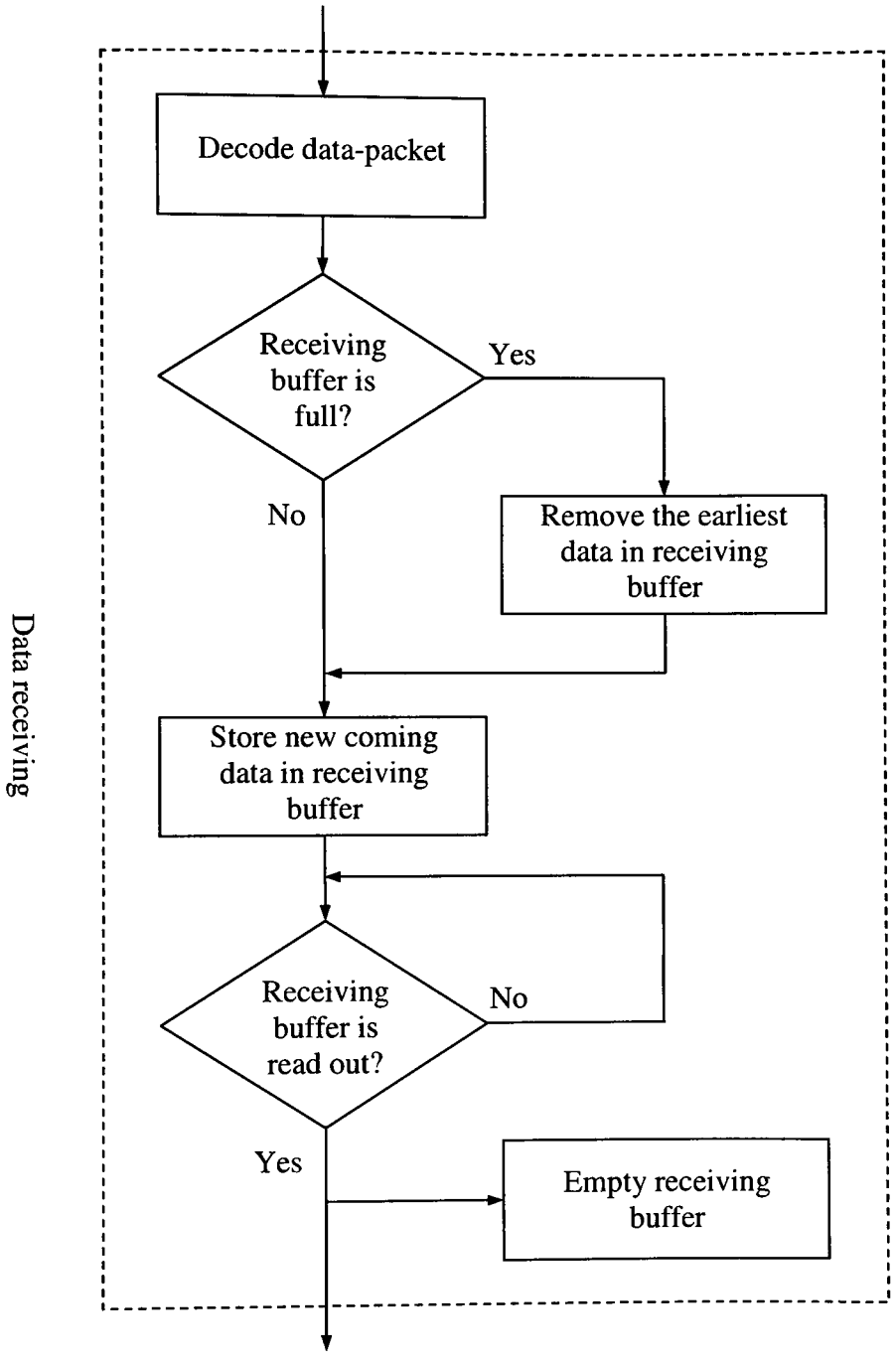



Figure 6.5: Data receiving.

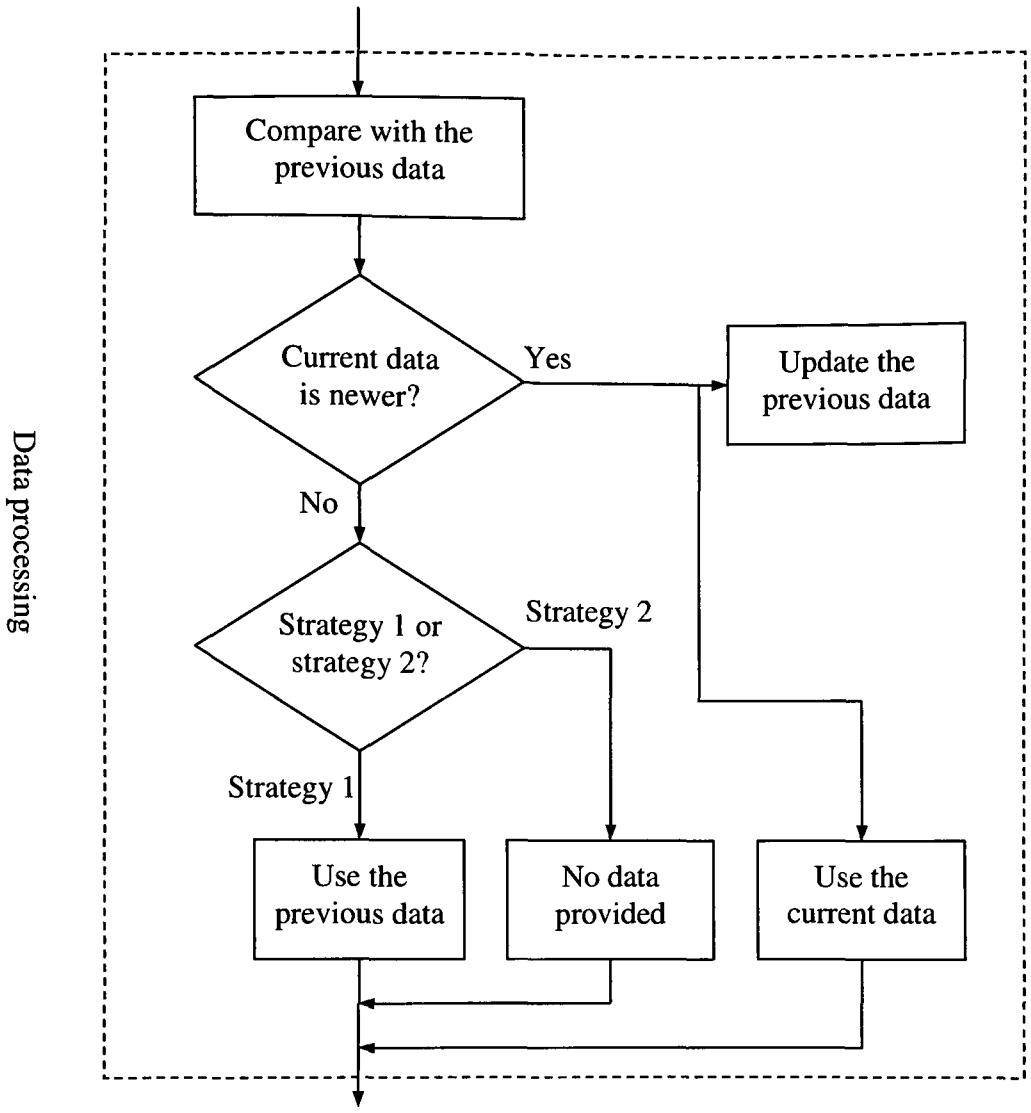



Figure 6.6: Data processing for Case1, Case2 and Case 3.

- not to provide any data, i.e., the controller or the actuator taking no action (Strategy 2).

See procedure flowchart in Figure 6.6. Strategy 1 implies an automatic increment of one step of the networked-induced delay; while Strategy 2, if applied to the controller side, means a dropout in controller-to-actuator network channel in future, and if applied to the controller side, it means a temporary unforced control on the plant.

**Case3:** there is no data obtained from data-receiving unit at the current time instant. Clearly, this is a special case of no newer data obtained in Case2.

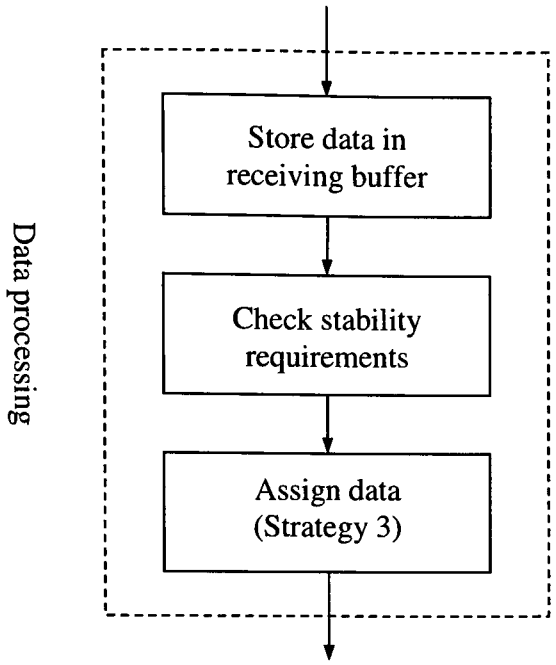



Figure 6.7: Data processing for Case4.

**Case4:** on the other hand, based on the delay-assignment strategy discussed in Chapter 4, data used in the controller or the actuator is always

- to be assigned (Strategy 3)

from a storing buffer in data-processing unit according to the stability consideration of the closed-loop system, in despite of whether or not there is data obtained or whether or not the obtained data is the latest. The storing-buffer needs to keep obtained data within a period of time (buffer size). Referring to Chapter 4, the shortest time period, $\sigma_u + \varsigma + 1$, can make sure that there is at least one data in the buffer. This strategy is shown in Figure 6.7.

### 6.2.4 Plant interface

The plant interface, which generally includes configurations of an A/D channel for sensor measurement and a D/A channel for applying the control signal to an actuator, is completely hardware-related. This means it is not considered when doing the simulation study. Under the NCS implementation structure presented in this chapter, plant interface units have no difference compared to the traditional control

system.

When doing simulation study, the plant usually requires a much smaller calculation step compared to the system sampling period. Thus, between the plant and the rest part of the system, some rate-transmission units are needed, which can be treated as one kind of plant interface.

## 6.3 Development of NCS Toolbox

MATLAB and Simulink provide a good environment to quickly implement the simulation of a control system. Together with Real-time Workshop, the system can also be implemented for real-time applications. For the NCS, as discussed in the above sections, all issues introduced by the network have been characterized as some relatively standardized procedures. These procedures are easily realized using MATLAB codes (standard C codes), so that it is convenient to use them for simulation and real-time implementation of general NCSs.

The author has developed a MATLAB/Simulink-based "Networked Control System Toolbox". Its overall structure is shown in Fig 6.8. NCS Toolbox is compatible for MATLAB 6.5 or higher versions. With the sole NCS Toolbox, users can build their NCS simulation model just focusing on the design of new controllers. At this stage, when doing real-time implementation, the configurations of the network interface and the plant interface are subject to corresponding hardwares of NetCon embedded system (see Chapter 5 for details) which has enough input-output channels and high speed and can be widely applied to practical control systems. Two versions of NetCon systems, ARM 7 and ARM 9, are supported in current NCS Toolbox.

### 6.3.1 Main contents

According to the analysis in the previous sections, NCS Toolbox is correspondingly classified into four parts: Network Simulation, Network Interface, Plant Interface, Control Schemes. In order for users to familiarize themselves with NCS Toolbox quickly, some Demos are also included.

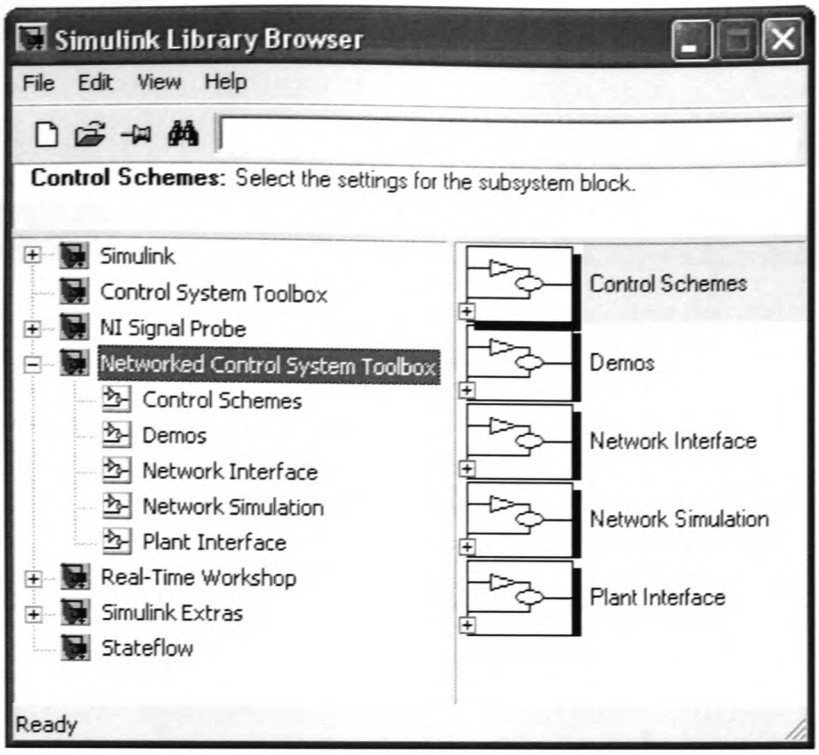



Figure 6.8: Networked Control System Toolbox.

In Network Simulation part (see Figure 6.9), two blocks, "Random delay" and "Markov delay", are developed to simulate networks with random transmission delay and Markov-chain delay, respectively. From the block interface, user should specify the lower and the upper bound of the delay, the data dropout rate and the Markov-chain matrix. A "Delay estimator" block is also comprised in this part. It calculates the network-induced delay. This is especially important for NPC control scheme where the accurate value of network-induced delay is required for generating control prediction sequence or choosing the exact control signal to compensate for the delay.

Network Interface (see Figure 6.10) provides two categories of data sending/receiving blocks for simulation and real-time application, respectively. Here UDP network protocols are adopted for data transferring. Except the target/source IP and Socket address, user also needs to specify the number of data which determines the size of packet transferring in the network.

In the Plant Interface (see Figure 6.11), various input-output types, for example AD, DA, PWM and PIO, are available. Each type might also have several channels.

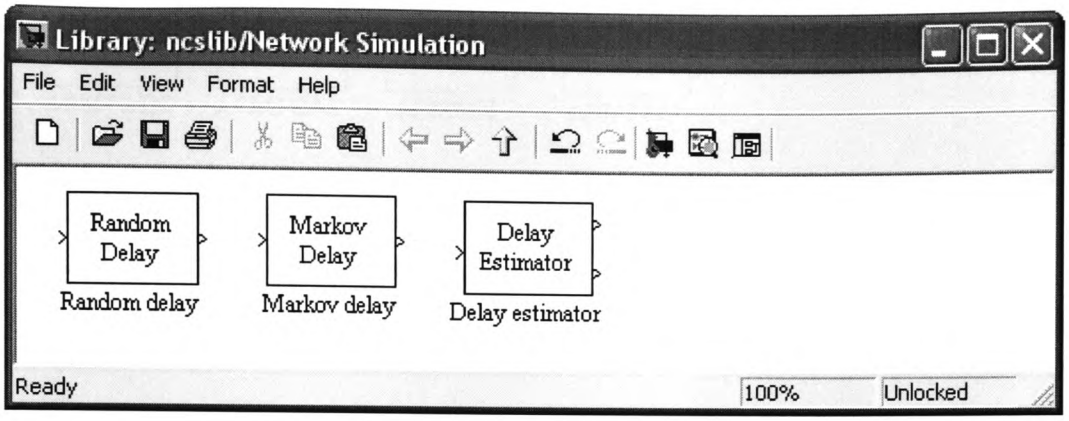



Figure 6.9: NCS Toolbox: Network Simulation.

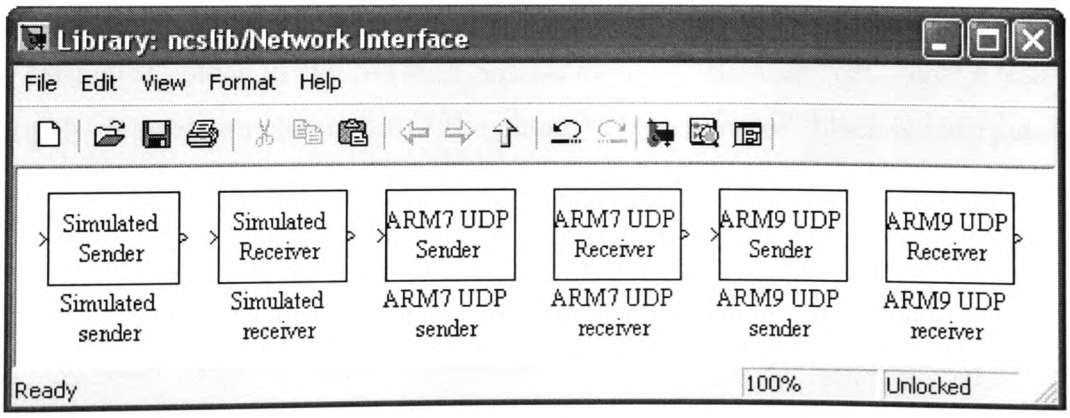



Figure 6.10: NCS Toolbox: Network Interface.

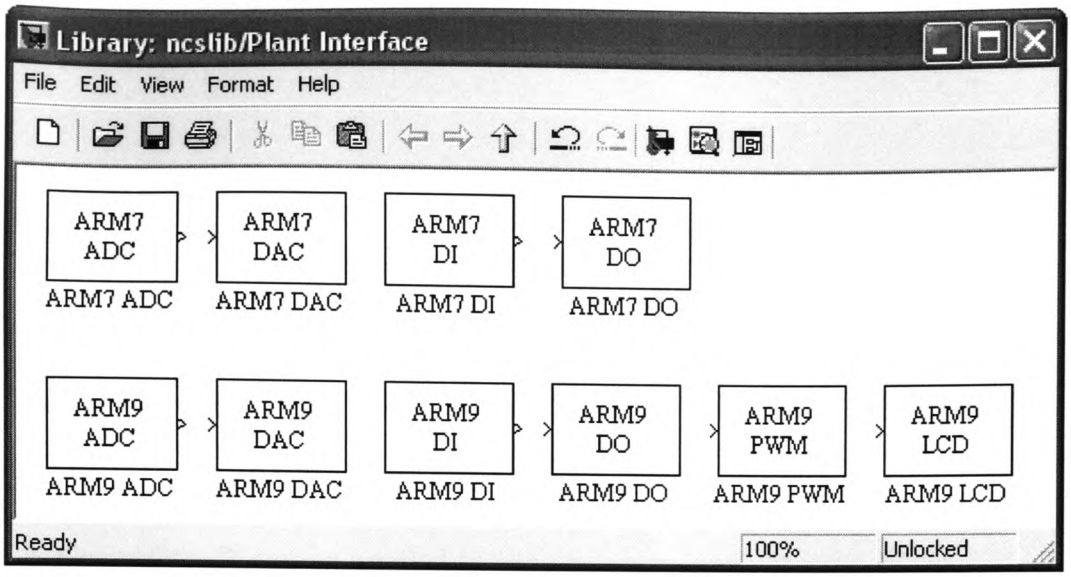



Figure 6.11: NCS Toolbox: Plant Interface.

so the user needs to specify which channel is to be used. Related program codes about Network Interface and Plant Interface were made by author's research unit. Here they are encapsulated in the corresponding toolbox functions.

In Control Schemes part (see Figure 6.12), NPC controllers are designed. It consists of "GPC predictor" block and "MPC predictor" block which use GPC (general predictive control) and MPC (model predictive control) methods to generate the control prediction sequence, respectively. In order to compensate for the network-induced delay, "Delay compensator" block is also presented, which is used with "Delay estimator" block in the Network Simulation part. Because NPC control scheme strongly depends on the model of the plant, "RLS estimator" block is incorporated in this part. It online modifies the model parameters using RLS (recursive least square) method. Another block in this part is "Data processing" in which user can choose different strategies discussed in Section 6.2.3.

Demos include some typical examples to demonstrate how to build a NCS model in MATLAB for both simulation and real-time implementation using NCS Toolbox.

### 6.3.2 Features

This NCS Toolbox has the following features:

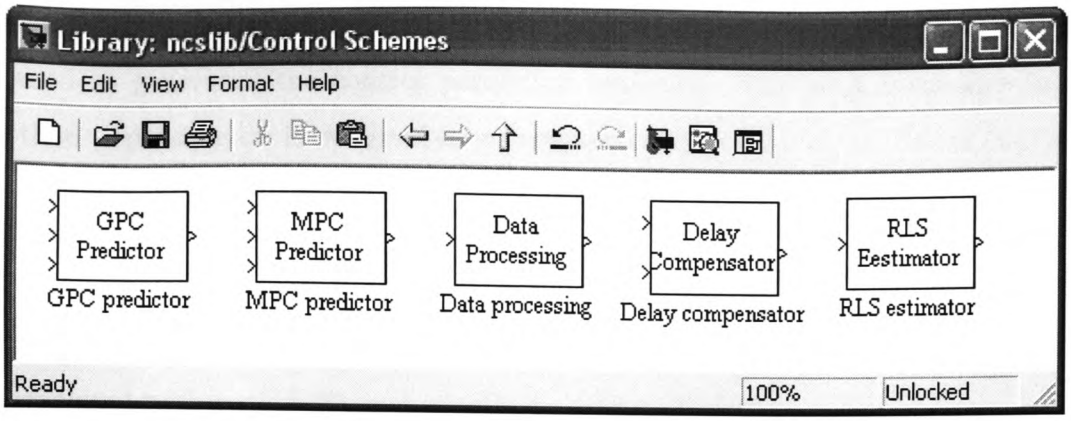



Figure 6.12: NCS Toolbox: Control Scheme.

- Generality. It can be applied to the implementation of most NCSs.
- Integration of simulation and real-time implementation. With the use of Real-Time Workshop, the simulation implementation can be downloaded to an embedded hardware system seamlessly to fulfill real-time implementation.
- Visual configuration of blocks. Initialization and parameters configuration of blocks are fulfilled by visual operation rather than writing codes.
- Error pre-detection. When configuring blocks, some basic errors will be detected immediately and proper prompts are also displayed.
- Help documents and demonstrations. Necessary help documents are affixed to each blocks. Some demonstrations of how to use the toolbox is also provided.
- Openness. This toolbox is open for future improvements and extension.

## 6.4 Examples

In this section, one demo in NCS Toolbox are presented here to show the convenience and effectiveness of using this toolbox to setup and implementation a NCS.

The control plant studied here is a servo system (Liu et al., 2006, 2007b) which has a transfer function model

$$\frac{B(z^{-1})}{A(z^{-1})} = \frac{0.05409z^{-2} + 0.115z^{-3} + 0.001z^{-4}}{1 - 1.12z^{-1} - 0.213z^{-2} + 0.335z^{-3}}$$

with sampling period 0.04s. NPC control scheme is taken here and MPC method is used to generated the control prediction sequence. The basic controller MPC method depending on is designed as a P-type controller

$$\frac{D(z^{-1})}{C(z^{-1})} = 0.5$$

which can guarantee the network-free system stable. The input reference is a square wave signal with period 20s and amplitude $-40 \sim 40$ degree.

Two Arm 9-based NetCon systems are used to construct the real-time implementation. One is for the local data acquisition with the servo system, and the other one is for the remote controller. Both NetCon systems communicate with each other via Internet.

### 6.4.1 Simulation implementation

The simulation model is shown in Figure 6.13. In this model, the network adopts "random delay" type with lower bound 0 and upper bound 10 steps, and data dropout rate 0.1. The "data processing" block is to continue providing the previous data if current data is not the latest (Strategy 1). The above plant model and controller should be used to configure "MPC predictor" block. Here it is assumed that unilateral networked-transmission delay is not available, so "Delay estimator" block has to measure the round-trip delay and half of it is treated as the bilateral delay. The simulation result is shown in Figure 6.14, where the thin line is the reference and the bold line the output. Compared the result to the that of the local control shown in Figure 6.15, it can be seen that the NPC method compensates the networked-induced delay very well.

### 6.4.2 Real-time implementation

The real-time implementation model is divided into two pars: controller side and plant side. It is very easy to build from the simulation model just by removing network simulation block in Figure 6.13 and replacing "simulated sender/receiver" blocks with "Arm9 UDP send/receiver" blocks. The plant model is also substituted

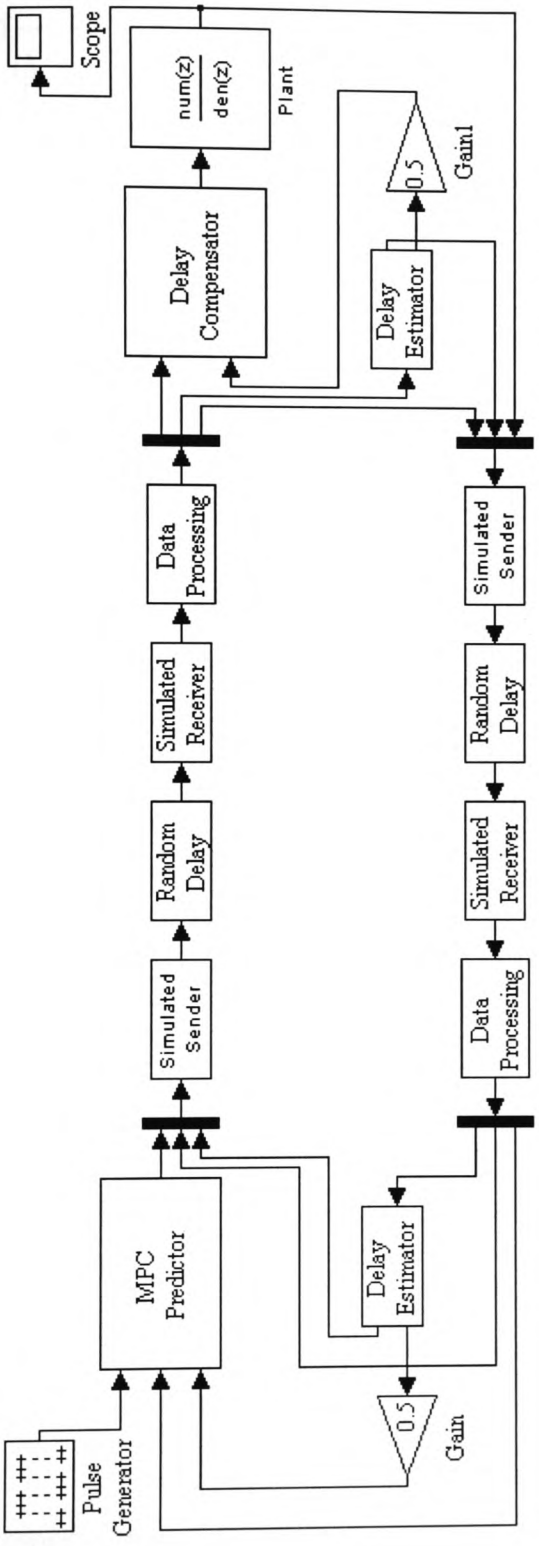



Figure 6.13: Simulation model of networked servo system built using NCS Toolbox.

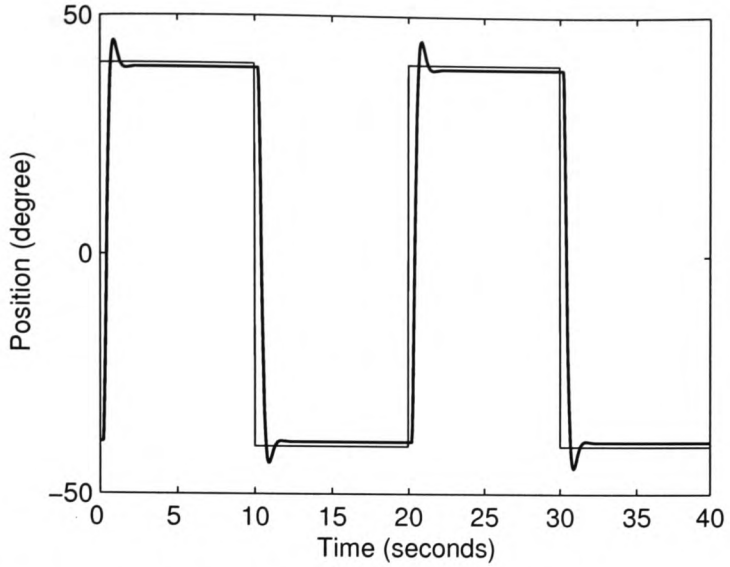



Figure 6.14: Response of simulation implementation of networked servo system.

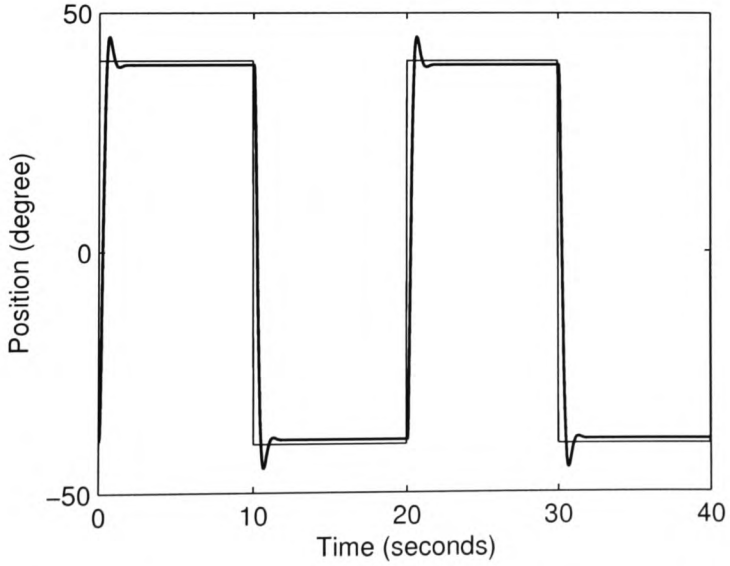



Figure 6.15: Response of simulation implementation of local servo system.

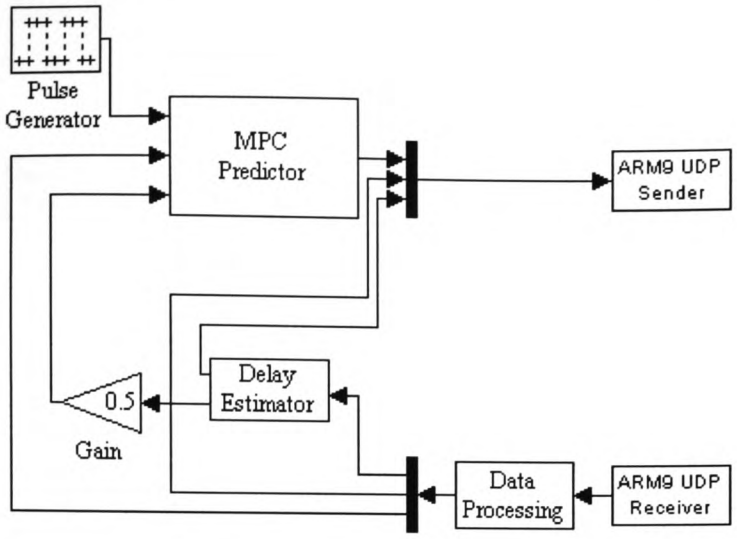



Figure 6.16: Real-time implementation model of networked servo system built using NCS Toolbox: controller side.

using input-output blocks for the practical plant. See Figure 6.16 and Figure 6.17. In this experiment, the servo plant is located in the University of Glamorgan, U.K., with IP address 193.63.131.170, and the remote controller is at the Chinese Academy of Sciences, China, with IP address 159.226.20.109. The system responses for networked and local cases are shown in Figure 6.18 and Figure 6.19, respectively, where the thin line is the reference and the bold line the output. It can be seen that the NPC method gives an accept result.

## 6.5 Summary

This chapter has discussed the development of a MATLAB/Simulink-based NCS toolbox for the implementation of NCSs. With this toolbox, users can easily build up a simulation model for their problems, so that they can concentrate on the study of more advanced control schemes. Real-time implementation of NCSs is also very convenient from its simulation model. In fact, it includes only small modification of several blocks.

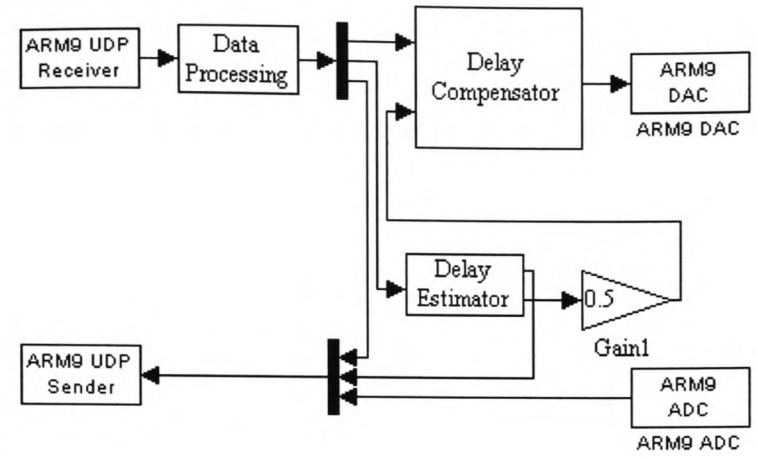



Figure 6.17: Real-time implementation model of networked servo system built using NCS Toolbox: plant side.

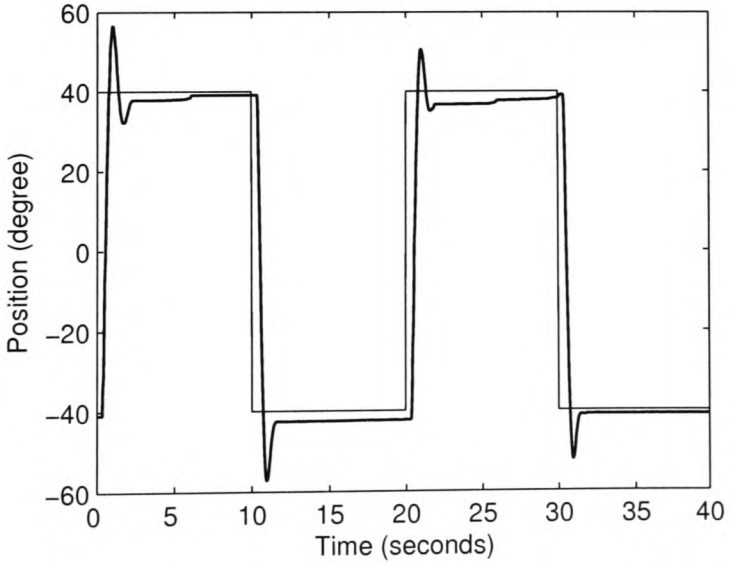



Figure 6.18: Response of real-time implementation of networked servo system.

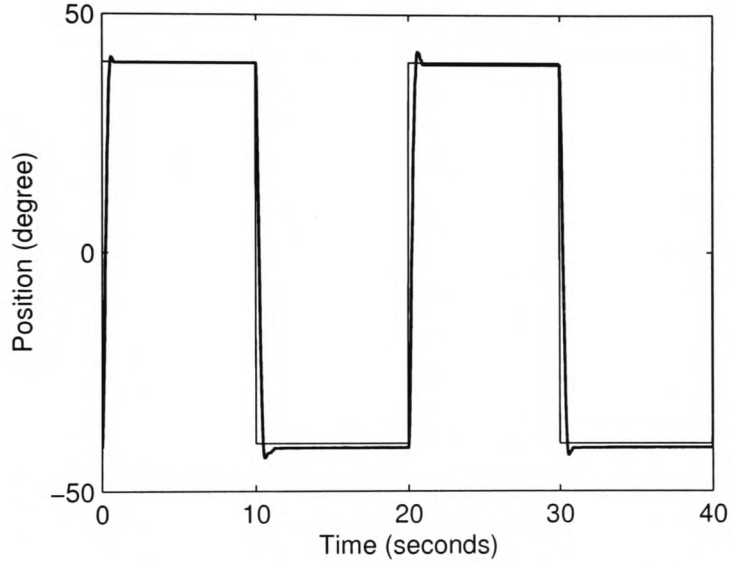



Figure 6.19: Response of real-time implementation of local servo system.

# Chapter 7

# Conclusion

In this concluding chapter, the major contribution of this thesis are summarized and some possible topics for future research are outlined.

## 7.1 Major contributions

This thesis dealt with the control and implementation of time-delay systems and networked control systems. The latter is a special case of the former with random network transmission delays, data-packet dropouts, asynchronous clocks and so on.

In general time-delay systems with constant delays, the control limits of a PI controller for IPDT were derived in frequency-domain. These limits include the stability region of control parameters, gain/phase margins and ISE performance for setpoint tracking and disturbance rejection. Furthermore, the above control limits under single tuning-parameter PI control and 2-DOF PI control structure were also obtained.

About the implementation of distributed delay which is often included in various modified Smith predictor schemes, a method to estimate the minimal implementation steps was proposed based on the small gain theorem, where the impact of the other parts of the controller and the different realization of the plant on the estimation was also studied to improve the estimation. The result is somewhat conservative in some cases, but at least it provides a guidance to choose the practical implementation steps.

Aiming at the strictness of the stability condition in NPCSs where all subsystems of the closed-loop switched system have to share a common positive definite matrix, an improvement was proposed so that the network-induced delay can be assigned. This corresponds to the assignment of subsystem. Hence, it is possible to relax the stability condition. The average dwell time method was employed here to design the assignment.

Control of MagLev system is a challenge, especially over network. A MagLev test rig was built as the implementation platform. Then the NPC method based on different models (feedback linearization model, direct linearization model and online parameter-adjusting model) was tested on this test rig to show its advantage over other methods without using prediction.

In order to implement NCSs as conveniently as to implement local control systems, a MATLAB/Simulink based NCS toolbox was developed, which involves both simulation and real-time functions of the general issues induced by the network.

## 7.2 Future work

Further to the work in this thesis, some possible directions for future work are outlined.

(i) The methods used to derive the control limits of PI controllers for IPDTs can be extended to the first-order processes plus time delay and other general time-delay systems.

(ii) The estimation of the minimal implementation steps of distributed delay is a little conservative in some cases mainly due to the small gain theorem. A challenging work is to seek other robust stability criteria to obtain less conservative results.

(iii) Networked predictive control strongly depends on the plant model because it is directly used to construct the controller. Any inaccuracy of the model will cause significantly improper control. So the study of the robustness of NPCSs is a future issue.

(iv) Multi sensors and multi actuators are usual in the general networked control systems. They often have different sampling periods and different clocks. Coordinate these sensors and actuators is a big challenge.

(v) When using GPC method to generate the future control prediction sequence in NPCS, the feedback channel and forward channel network-induced delay have different effects on the system performance. It is necessary to study more about this and to derive the minimal allowable delay to guarantee the system stability in future.

(vi) For the studied MagLev system, the current control method need to be improved to bear a larger delay.

(vii) Add functions about event-driven controller and actuator to NCS toolbox. Keep updating NCS toolbox with more advanced control schemes.

# Bibliography

Åström, K. J., Hang, C. C., Lim, B. C., 1994. A new smith predictor for controlling a process with an integrator and long dead-time. IEEE Trans. Automat. Contr. 39 (2), 343–345.

Barie, W., Chiasson, J., 1996. Linear and nonlinear state-space controllers for magnetic levitation. Int. J. Syst. Sci. 27 (11), 1153–1163.

Battilotti, S., Feb. 2008. Control over a communication channel with random noise and delays. Automatica 44 (2), 348–360.

Bellman, R., Cooke, K. L., 1963. Differential-difference equations. Academic Press, New York.

BYTRONIC, 2006. Magnetic levitation system: user's manual. Bytronic Ltd.

Cervin, A., Henriksson, D., Lincoln, B., Eker, J., Arzen, K.-E., Jun. 2003. How does control timeing affect performance? IEEE Control Syst. Mag. 23, 16–30.

Charara, A., Miras, J. D., Caron, B., Sep. 1996. Nonlinear control of a magnetic levitation system without premagnetization. IEEE Trans. Contr. Syst. Technol. 4, 513–523.

Chen, J., 1995. On computing the maximal delay intervals for stability of linear delay systems. IEEE Trans. Automat. Contr. 40 (9), 1089–1093.

Chiasson, J., 1989. Ee3648 nonlinear systems class notes. Tech. rep., University of Pittsburgh, Pittsburgh, Pensylvania.

Chiasson, J., Abdallah, C. T., 2001. Robust stability of time delay systems: Theory. In: Proc. 3rd IFAC Workshop on Time Delay Systems. Santa Fe, NM.

Chidambaram, M., Sree, R. P., 2003. A simple method of tuning PID controllers for integrator/dead time processes. Comput. Chem. Eng. 27, 211–215.

Cho, D., Kato, Y., Spilman, D., 1993. Sliding mode and classical control of magnetic levitation systems. IEEE Control Syst. Mag., 42–48.

Dahlen, N. J., 1985. Magnetic active suspension and isolation. Master's thesis, Dept. Mech. Eng., Massachusetts Inst. Technol., Cambridge, MA.

Downer, J. R., 1980. Analysis of single axis magnetic suspension systems. Master's thesis, Dept. Mech. Eng., Massachusetts Inst. Technol., Cambridge, MA.

Dussaux, M., 1990. The industrial applications of the active magnetic bearings technology. In: Proc. 2nd Int. Symp. Magnetic Bearings. pp. 33–38.

Elia, N., Mitter, S. K., 2001. Stabilization of linear systems with limited information. IEEE Trans. Automat. Contr. 46 (9), 1384–1400.

Engelborghs, K., Dambrine, M., Roose, D., Feb. 2001. Limitations of a class of stabilization methods for delay systems. IEEE Trans. Automat. Contr. 46, 336–339.

Ge, S. S., Hong, F., Lee, T. H., 2003. Adaptive neural network control of nonlinear systems with unknown time delays. IEEE Trans. Automat. Contr. 48 (11), 2004–2010.

Gu, K., Niculescu, S.-I., 2003. Survey on recent results in the stability and control of time-delay systems. J. Dyn. Syst. Meas. Contr. 125, 158–165.

Hajjaji, A. E., Ouladsine, M., Aug. 2001. Modelling and nonlinear control of magnetic levitation systems. IEEE Trans. Ind. Electron. 48 (4), 831–838.

Hale, J. K., Verduyn-Lunel, S. M., 1993. Introduction to functional differential equations. In: Applied Mathematical Sciences,. Vol. 99. Springer, New York.

He, Y., Wang, Q.-G., Lin, C., Wu, M., Feb. 2007. Delay-range-dependent stability for systems with time-varying delay. Automatica 43 (2), 371–376.

He, Y., Wu, M., She, J.-H., Jul. 2006. Delay-dependent exponential stability of delayed neural networks with time-varying delay. IEEE Trans. Circuits Syst. II, Exp. Briefs 53 (7), 553–557.

Hespanha, J., McLaughlin, M., Sukhatme, G., Akbarian, M., Garg, R., Zhu, W., 2002a. Haptic collaboration over the internet. Touch in Virtual Environments: Haptics and the Design of Interactive Systems. Prentice Hall.

Hespanha, J., Ortega, A., Vasudevan, L., 2002b. Towards the control of linear systems with minimum bit-rate. In: Proc. Int. Symp. Mathematical Theory of Networks and Syst.

Hespanha, J. P., Sep. 2006. Modelling and analysis of stochastic hybrid systems. IEE Proc. Control Theory Appl. 153 (5), 520–535.

Hespanha, J. P., Morse, A. S., Dec. 1999. Stability of switched systems with average dwell-time. In: Proc. 38th IEEE Conf. Decision and Control. Phoenix, USA.

Hikichi, K., Morino, H., Arimoto, I., Sezaki, K., Yasuda, Y., 2002. The evaluation of delay jitter for haptics collaboration over the Internet. In: Proc. IEEE Conf. Global Telecommunications, GLOBECOM'02. Vol. 2.

Hong, S., 1995. Scheduling algorithm of data sampling times in the integrated-communication and control systems. IEEE Trans. Contr. Syst. Technol. 3 (2), 225–230.

Hu, W., Liu, G.-P., Rees, D., Jun. 2007. Event-driven networked predictive control. IEEE Trans. Ind. Electron. 54 (3), 1603–1613.

Isidori, A., 1989. Nonlinear control systems. Springer-Verlag, New York.

Jankovic, M., 2001. Control lyapunov-razumikhin functions and robust stabilization of time delay systems. IEEE Trans. Automat. Contr. 46 (7), 1048–1060.

Johannessen, S., 2004. Time synchronization in a local area network. IEEE Control Syst. Mag. 24 (2), 61–69.

Joo, A. J., Seo, J. H., Jan. 1997. Design and analysis of the nonlinear feedback linearizing control for an electromagnetic suspension system. IEEE Trans. Contr. Syst. Technol. 5, 135–144.

Kharitonov, V. L., 1999. Robust stability analysis of time delay systems: A survey. Annual Review in Control 23, 185–196.

Kim, C. Y., Kim, K. H., Jun. 1994. Gain scheduling control of magnetic suspension systems. In: Proc. Amer. Contr. Conf. pp. 3127–3131.

Kim, D., Lee, Y., Kwon, W., Park, H., 2003. Maximum allowable delay bounds of networked control systems. Control Eng. Pract. 11 (11), 1301–1313.

Kim, W.-J., Ji, K., Ambike, A., Sep. 2006a. Networked real-time control strategy dealing with stochastic time delays and packet losses. ASME Trans. J. Dyn. Syst. Meas. Control 128, 681–685.

Kim, W.-J., Ji, K., Ambike, A., Jul. 2006b. Real-time operating environment for networked control systems. IEEE Trans. Automat. Sci. Eng. 3 (3), 287–296.

Kolmanovskii, V. B., Niculescu, S.-I., Gu, K., 1999. Delay effects on stability: a survey. In: Proc. 38th IEEE Conf. Control and Decision. pp. 1993–1998.

Kolmanovskii, V. B., Nosov, V. R., 1986. Stability of functional differential equations. Academic Press, London.

Krasovskii, N. N., 1963. Stability of motion. Stanford University Press, Stanford CA, USA.

Limbert, D. A., Richardson, H. H., Wormley, D. N., 1990. Controlled characteristics of ferromagnetic vehicle suspension providing simultaneous lift and guidance. ASME Trans. J. Dyn. Syst. Meas. Control 101, 217–222.

Lin, F.-J., Teng, L.-T., Shieh, P.-H., May 2007. Intelligent adaptive backstepping control system for magnetic levitation apparatus. IEEE Trans. Magnetics 43 (5), 2009–2018.

Lin, H., Zhai, G., Antsaklis, P. J., Dec. 2003. Robust stability and disturbance attenuation analysis of a class of networked control systems. In: Proc. 42nd IEEE Conf. Decision and Control. Hawaii, USA.

Liu, G.-P., Mu, J., Rees, D., 2004. Networked predictive control of systems with random communication delays. In: Proc. UKACC Int. Conf. Control.

Liu, G.-P., Mu, J., Rees, D., Chai, S., Apr. 2006. Design and stability analysis of networked control systems with random communication time delay using the modified MPC. Int. J. Control 79 (4), 288–297.

Liu, G.-P., Rees, D., Chai, S., Nie, X., Feb. 2005. Design, simulation and implementation of networked predictive control systems. Meas. Control 38 (1), 17–21.

Liu, G.-P., Xia, Y., Chen, J., Rees, D., Hu, W., 2007a. Networked Predictive Control of Systems With Random Network Delays in Both Forward and Feedback Channels. IEEE Trans. Ind. Electron. 54 (3), 1282–1297.

Liu, G.-P., Xia, Y., Rees, D., Hu, W., Mar. 2007b. Design and stability criteria of networked predictive control systems with random network delay in the feedback channel. IEEE Trans. Syst., Man, Cybern. C, Appl. Rev. 37 (2), 173–184.

Lu, X., Yang, Y.-W., Wang, Q.-G., Zheng, W.-X., 2005. A double two-degree-of-freedom control scheme for improved control of unstable delay processes. J. Process Control 15 (5), 605–614.

Luyben, W. L., 1996. Tuning proportional-integral-derivative controllers for integrator/deadtime processes. Ind. Eng. Chem. Res. 35, 3480–3483.

Manitius, A. Z., Olbrot, A. W., 1979. Finite spectrum assignment probolem for systems with delays. IEEE Trans. Automat. Contr. 24 (4), 541–553.

Meinsma, G., Zwart, H., Feb. 2000. On $H^\infty$ control for dead-time systems. IEEE Trans. Automat. Contr. 45, 272–285.

Meng, C., Wang, T., Chou, W., Luan, S., Zhang, Y., Tian, Z., 2004. Remote surgery case: robot-assisted teleneurosurgery. In: Proc. IEEE Int. Conf. Robotics and Automation, ICRA'04. Vol. 1.

Mirkin, L., Sep. 2003a. Are distributed-delay control laws intrinsically unapproximable? In: Proc. 4th IFAC workshop on time-delay systems, TDS'03. Rocquencourt, France.

Mirkin, L., Apr. 2003b. On the extraction of dead-time controllers and estimators from delay-free parametrizations. IEEE Trans. Automat. Contr. 48, 543–553.

Mirkin, L., 2004. On the approximation of distributed-delay control laws. Systems & Control letters 51, 331–342.

Mirkin, L., Raskin, N., 2003. Every stabilizing dead-time controller has an observer-predictor-based structure. Automatica 39 (10), 1747–1754.

Montestruque, L., Antsaklis, P., 2003. Stochastic stability for model-based networked control systems. In: Proc. American Control Conf. Vol. 5.

Montestruque, L., Antsaklis, P., Sep. 2004. Stability of model-based networked control systems with time-varying transmission times. IEEE Trans. Automat. Contr. 49 (9), 1562–1572.

Nair, G., Evans, R., 2003. Exponential stabilisability of finite-dimensional linear systems with limited data rates. Automatica 39 (4), 585–593.

Nemoto, Y., Hamamoto, N., Suzuki, R., Ikegami, T., Hashimoto, Y., Ide, T., Ohta, K., Mansfield, G., Kato, N., 1997. Construction and Utilization Experiment of Multimedia Education System Using Satellite ETS-V and Internet. IEICE Trans. on Information and Systems 80 (2), 162–169.

Niculescu, S.-I., 2001. Delay effects on stability: a robust control approach. Springer-Verlag, Heidelberg, Germany.

Nijmeijer, H., Schaft, A. V. D., 1990. Nonlinear dynamical control systems. Springer-Verlag, New York.

Nilsson, J., 1998. Real-time control systems with delays. Department of Automatic Control, Lund Institute of Technology, Lund, Sweden.

Nilsson, J., Bernhardsson, B., Wittenmark, B., 1998. Stochastic Analysis and Control of Real-time Systems with Random Time Delays. Automatica 34 (1), 57–64.

Normey-Rico, J. E., Camacho, E. F., 1999. Robust tuning of dead-time compensators for processes with an integrator and long deat-time. IEEE Trans. Automat. Contr. 44 (8), 1597–1603.

Ogren, P., Fiorelli, E., Leonard, N., 2004. Cooperative control of mobile sensor networks: Adaptive gradient climbing in a distributed environment. IEEE Trans. Automat. Contr. 49 (8), 1292–1302.

Olgac, N., Sipahi, R., 2002. An exact method for the stability analysis of time-delay lti systems. IEEE Trans. Automat. Contr. 47 (5), 793–797.

Overstreet, J., Tzes, A., 1999. An Internet-based real-time control engineering laboratory. IEEE Control Syst. Mag. 19 (5), 19–34.

Palmor, Z. J., 1996. The Control Handbook, S. Levine Edition. CRC Press, Boca Raton, FL, Ch. Time-delay compensation — Smith predictor and its modifications, pp. 224–237.

Park, H. S., Kim, Y. H., Kim, D.-S., Kwon, W. H., May 2002. A scheduling method for network-based control systems. IEEE Trans. Contr. Syst. Technol. 10 (3), 318–330.

Pontryagin, L. S., 1955. On the zeros of some elementary transcendental functions. Am. Math. Soc. Trans., 95–110.

Poulin, E., Pomerleau, A., 1999. PI settings for integrating processes based on ultimate cycle information. IEEE Trans. Contr. Syst. Technol. 7 (4), 509–511.

RAY, A., 1987. Performance evaluation of medium access control protocols for distributed digital avionics. ASME Trans. J. Dyn. Syst. Meas. Control 109, 370–377.

Richard, J.-P., 2003. Time-delay systems: An overview of some recent advances and open problems. Automatica 39 (10), 1667–1694.

Rivera, D. E., Morari, M., Skogestad, S., 1986. Internal model control 4: PID controller design. Ind. Eng. Chem. Process Des. Dev. 25, 252–265.

Santos, O., Mondie, S., 2000. Control laws involving distributed time delays: Robustness of the implementation. In: Proc. Amer. Control Conf. pp. 2479–2480.

Seiler, P., Sengupta, R., 2001. Analysis of communication losses in vehicle control problems. In: Proc. American Control Conf. Vol. 2.

Seiler, P., Sengupta, R., 2005. An$ H_infty$ Approach to Networked Control. IEEE Trans. Automat. Contr. 50 (3), 356–364.

Shirmohammadi, S., Woo, N., 2004. Evaluating Decorators for Haptic Collaboration over Internet. In: Proc. IEEE Workshop on Haptic Audio Visual Environments and their Applications (IEEE HAVE'04), Ottawa, Canada. pp. 105–110.

Silva, G. J., Datta, A., Bhattacharyya, S. P., 2002. New results on the synthesis of PID controllers. IEEE Trans. Automat. Contr. 47 (2), 241–251.

Smith, O. J. M., 1957. Closer control of loops with dead time. Chem. Eng. Progress 53 (5), 217–219.

Strom, T., Dec. 1975. On logarithmic norms. SIAM J. Numerical Analysis 12 (6), 971–981.

Su, W., Qiu, L., Chen, J., 2003. Fundamental performance limitations in tracking sinusoidal signals. IEEE Trans. Automat. Contr. 48 (8), 1371–1380.

Sun, L., Ohmori, H., Sano, A., Aug. 1999. Direct closed-loop identification of magnetic suspension system. In: Proc. IEEE Int. Conf. Contr. Appli. Hawai.

Sun, X.-M., Zaho, J., Hill, D. J., Oct. 2006. Stability and $L_2$-gain analysis for switched delay: a delay-dependent method. Automatica 42 (10), 1769–1774.

Sun, Z., Ge, S.-S., 2004. Switched linear systems - control and design. New York: Srpinger.

Tatikonda, S., Mitter, S., 2004. Control under communication constraints. IEEE Trans. Automat. Contr. 49 (7), 1056–1068.

Taylor, K., Dalton, B., 2000. Internet robots: a new robotics niche. IEEE Rob. Autom Mag. 7 (1), 27–34.

Teng, F. C., 2002. Real-time control using matlab simulink. In: IEEE Int. Conf. Systems, Man, and Cybernetics. pp. 2697–2702.

Tipsuwan, Y., Chow, M.-Y., 2003. Control methodologies in networked control systems. Control Eng. Pract. 11, 1099–1111.

Trumpler, D. L., Olson, S. M., Subrahmanyan, P. K., Jul. 1997. Linearizing control of magnetic suspension systems. IEEE Trans. Contr. Syst. Technol. 5, 427–437.

Tyreus, B. D., Luyben, W. L., 1992. Tuning PI controllers for integrator/dead-time processes. Ind. Eng. Chem. Res. 31, 2625–2628.

Van Assche, V., Dambrine, M., Lafay, J. F., Richard, J. P., 1999. Some problems arising in the implementation of distributed-delay control laws. In: Proc. 38th IEEE Conf. Decision and Control. Phoenix, AZ, pp. 4668–4672.

Van Loan, C., Dec. 1977. The sensitivity of the matrix exponential. SIAM J. Numerical Analysis 14 (6), 971–981.

Visioli, A., 2001. Optimal tuning of PID controllers for integral and unstable processes. IEE Proc. Control Theory Appl. 148 (2), 180–184.

Walsh, G., Beldiman, O., Bushnell, L., 2001. Asymptotic behavior of nonlinear networked control systems. IEEE Trans. Automat. Contr. 46 (7), 1093–1097.

Walton, K., Marshall, J. E., 1987. Direct method for tds stability analysis. IEE Proc. Control Theory Appl. 134 (2), 101–107.

Wang, B., Liu, G.-P., Rees, D., 2007a. Networked predictive control systems with partially-assigned network delay. submitted to IEEE Trans. Circuits Syst. II, Exp. Briefs.

Wang, B., Rees, D., Zhong, Q., 2006. Control of integral processes with dead time. Part IV: various issues about PI controllers. IEE Proc. Control Theory Appl. 153 (3), 302–306.

Wang, B., Wang, R., Liu, G.-P., Rees, D., 2007b. Stability analysis of networked predictive control systems using an average dwell time method. submitted to IET Proc. Control Theory Appl.

Wang, B., Zhang, J. F., Zhong, Q.-C., Rees, D., 2007c. On minimal number of implementation steps of distributed delays in linear control laws. to be submitted.

Wang, L., Cluett, W. R., 1997. Tuning PID controllers for integrating processes. IEE Proc. Control Theory Appl. 144 (5), 385–392.

Wang, Q. G., Lee, T. H., Tan, K. K., 1999. Finite spectrum assignment for time-delay systems. Springer-Verlag, London, U.K.

Wang, R., Liu, G.-P., Wang, B., Rees, D., 2007d. L2-gain analysis for networked prdictive control systems based on switching method. submitted to IEEE Trans. Syst., Man, Cybern. C, Appl. Rev.

Watanabe, K., Jun. 1986. Finite spectrum assignmnet and observer for multivariable systems with commensurate delays. IEEE Trans. Automat. Contr. 31, 543–550.

Watanabe, K., Ito, M., 1981. A process-model control for linear systems with delay. IEEE Trans. Automat. Contr. 26 (6), 1261–1269.

Wong, W., Brockett, R., 1999. Systems with finite communication bandwidth constraints. II. Stabilization with limited information feedback. IEEE Trans. Automat. Contr. 44 (5), 1049–1053.

Yamamura, S., Yamaguchi, H., 1990. Electromagnetic levitation system by means of salient-pole type magnets coupled with laminated slotless rails. IEEE Trans. Vehicular Technol. 39, 83–87.

Yang, S., Tan, L., Chen, X., 2002a. Specification and architecture design for internet-based control systems. In: Proc. 26th Annual Int. Computer Software and Applications Conf., IEEE Computer Society, Oxford, UK, 2002b. pp. 75–80.

Yang, S.-H., Chen, X., Alty, J. L., Jun. 2003. Design issues and implementation of internet-based process control systems. Control Eng. Pract. 11 (6), 709–720.

Yang, T.-C., Jul. 2006. Networked control system: a brief survey. IEE Proc. Control Theory Appl. 153 (4), 403–412.

Yang, Z.-J., Miyazaki, K., Jin, C.-Z., Wada, K., 2002b. Physical parameter identification of a magnetic levitation system under a robust nonlinear controller. In: 15th Triennial World Congress. Barcelona, Spain.

Yang, Z.-J., Miyazaki, K., Kanae, S., Wada, K., Feb. 2004. Robust position control of a magnetic levitation system via dynamic surface control technique. IEEE Trans. Ind. Electron. 51 (1), 26–34.

Yoo, H., Ryu, H., Yoo, K., Kwon, O., 2002. Compensation of networked control systems using LMI-based delay-dependent optimization method. In: Proc. 41st SICE Annual Conf. Vol. 1.

Yue, D., Han, Q.-L., Nov. 2005. Delay-dependent robust $H_\infty$ controller design for uncertain descriptor systems with time-varying discrete and distributed delays. IEE Proc. Control Theory Appl. 152 (6), 628–638.

Yue, D., Han, Q.-L., Peng, C., Nov. 2004. State feedback controller design of networked control systems. IEEE Trans. Circuits Syst. II, Exp. Briefs 51 (11), 640–644.

Zhai, G., Hu, B., Yusuda, K., Michel, A., Nov. 2001. Disturbance attenuation properties of time-controlled switched systems. J. Franklin Inst. - Engineering and Applied Mathematics 338 (7), 765–779.

Zhang, W., Branicky, M. S., Phillips, S. M., Feb. 2001. Stability of networked control systems. IEEE Control Syst. Mag. 21 (1), 84–99.

Zhong, Q.-C., 2003a. Control of integral processes with dead time. Part III: Deadbeat disturbance response. IEEE Trans. Automat. Contr. 48 (1), 153–159.

Zhong, Q.-C., 2003b. Frequency domain solution to delay-type Nehari problem. Automatica 39 (3), 499–508.

Zhong, Q.-C., 2003c. $H^\infty$ control of dead-time systems based on a transformation. Automatica 39 (2), 361–366.

Zhong, Q.-C., Jun. 2003d. On standard $H^\infty$ control of processes with a single delay. IEEE Trans. Automat. Contr. 48, 1097–1103.

Zhong, Q.-C., Nov. 2004. On distributed delay in linear control laws. Part I: Discrete-delay implementations. IEEE Trans. Automat. Contr. 49 (11), 2074–2080.

Zhong, Q.-C., 2005a. On distributed delay in linear control laws. Part II: Rational implementation inspired from the $\delta$-operator. IEEE Trans. Automat. Contr. 50 (5), 729–734.

Zhong, Q.-C., Jul. 2005b. Rational implementation of distributed delay using extended bilinear transformations. In: Proc. 16th IFAC World Congress. Prague, Czech Rep.

Zhong, Q.-C., 2006. Robust Control of Time-Delay Systems. Springer, London, U.K.

Zhong, Q.-C., Li, H. X., 2002. Two-degree-of-freedom PID-type controller incorporating the Smith principle for processes with dead-time. Ind. Eng. Chem. Res. 41 (10), 2448–2454.

Zhong, Q.-C., Mirkin, L., 2002. Control of integral processes with dead-time. Part II: Quantitative analysis. IEE Proc. Control Theory Appl. 149 (4), 291–296.

Zhong, Q.-C., Normey-Rico, J. E., 2002. Control of integral processes with dead time. Part I: A disturbance observer-based 2DOF control scheme. IEE Proc. Control Theory Appl. 149 (4), 285–290.

Zhong, Q.-C., Weiss, G., 2004. A unified Smith predictor based on the spectral decomposition of the plant. Int. J. Control 77 (15), 1362–1371.

Zhou, K., Doyle, J. C., Glover, K., 1996. Robust and Optimal Control. Prentice-Hall, Englewood Cliffs, NJ.

Zhu, Q., Lu, G., Cao, J., Hu, S., 2005. Stability analysis of networked control systems with Markov delay. In: Int. Conf. Control and Automation, ICCA'05. Vol. 2.

# Appendix A

# Using of NCS Toolbox

In this appendix, how to NCS toolbox is briefly presented.

## A.1 Network simulation

Take the "Random Delay" function as the example. Its configuration is shown in Figure A.1. There are four parameters to be configured: Maximum channel delay, Minimum channel delay, Data packet dropout probability and Sample time. The delay vary between the specified maximum and minimum delays. This block outputs "Inf" if data dropout happens. Here the delay is the integer multiplication of the actual time delay (in seconds) w.r.t. the sample time.

## A.2 Network interface

Take the "ARM 9 UDP sender" and "ARM 9 UDP Receiver" functions as the example. Their configurations are shown in Figure A.2. There are four parameters to be configured: Target/Source IP address, Socket port, Number of Data and Sample time.

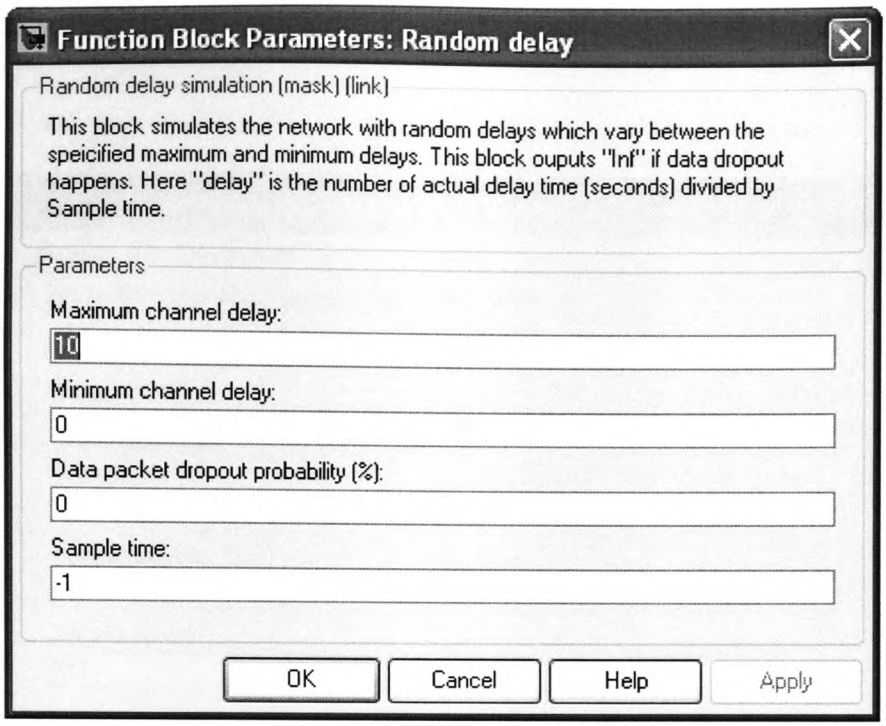



Figure A.1: Configuration of network simulation with random network transmission delay.

## A.3 Plant interface

Take the "ARM 9 ADC" and "ARM 9 DAC" functions as the example. Their configuration are shown in Figure A.3. There are two parameters to be configured: Port number and Sample time. User can choose an analog input port from channel 0-12 in ARM 9 ADC. Similarly, four channels 0-3 can be chosen as the analog output in ARM 9 DAC.

## A.4 Control scheme

Take the "MPC predictor" function as the example. Its configurations are shown in Figure A.4. For "MPC predictor", user needs to configure the transfer function model (Model numerator and Model denominator, Model delay), the controller transfer function (Controller numerator and control denominator), the forward channel maximum delay (Control prediction horizon), Maximum feedback channel delay and Sample time. The function has three input ports. Input 1 (from top to bottom)

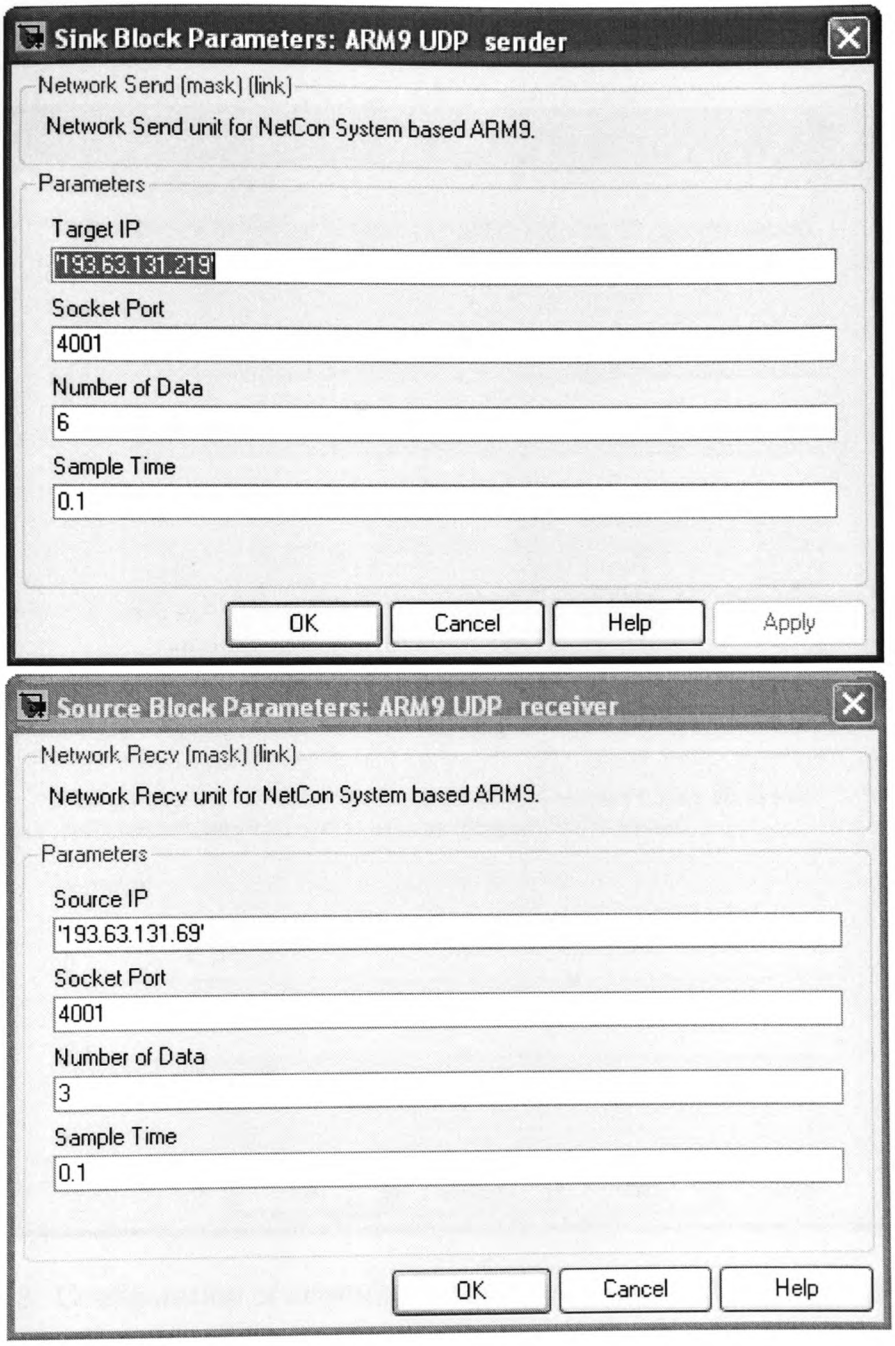



Figure A.2: Configuration of network interface with ARM 9 UDP sender and ARM 9 UDP receiver.

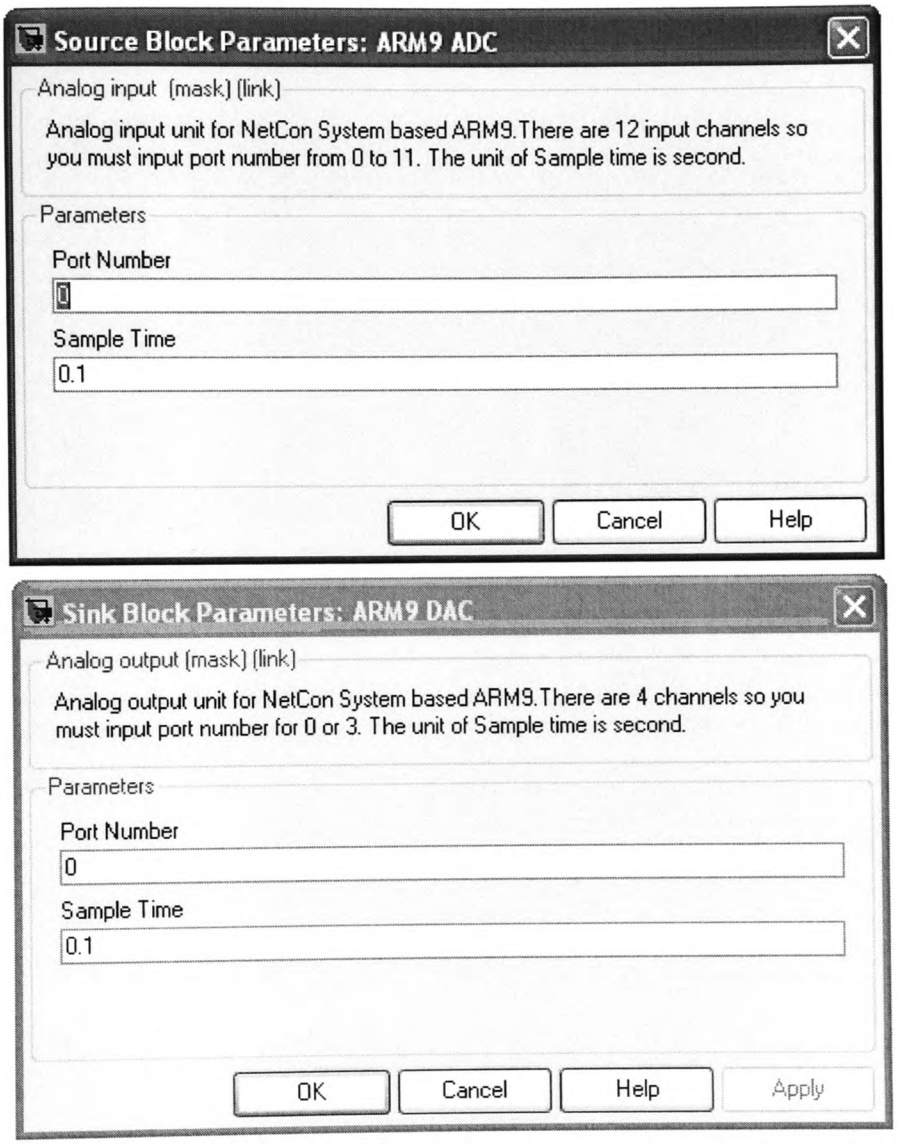



Figure A.3: Configuration of network interface with ARM 9 ADC and ARM 9 UDP DAC.

is the reference, Input 2 is the plant output feedback, and Input 3 is the feedback channel delay. This function generates the future control predictions up to the control prediction horizon, which is based on the past plant outputs up to the maximum feedback channel delay.

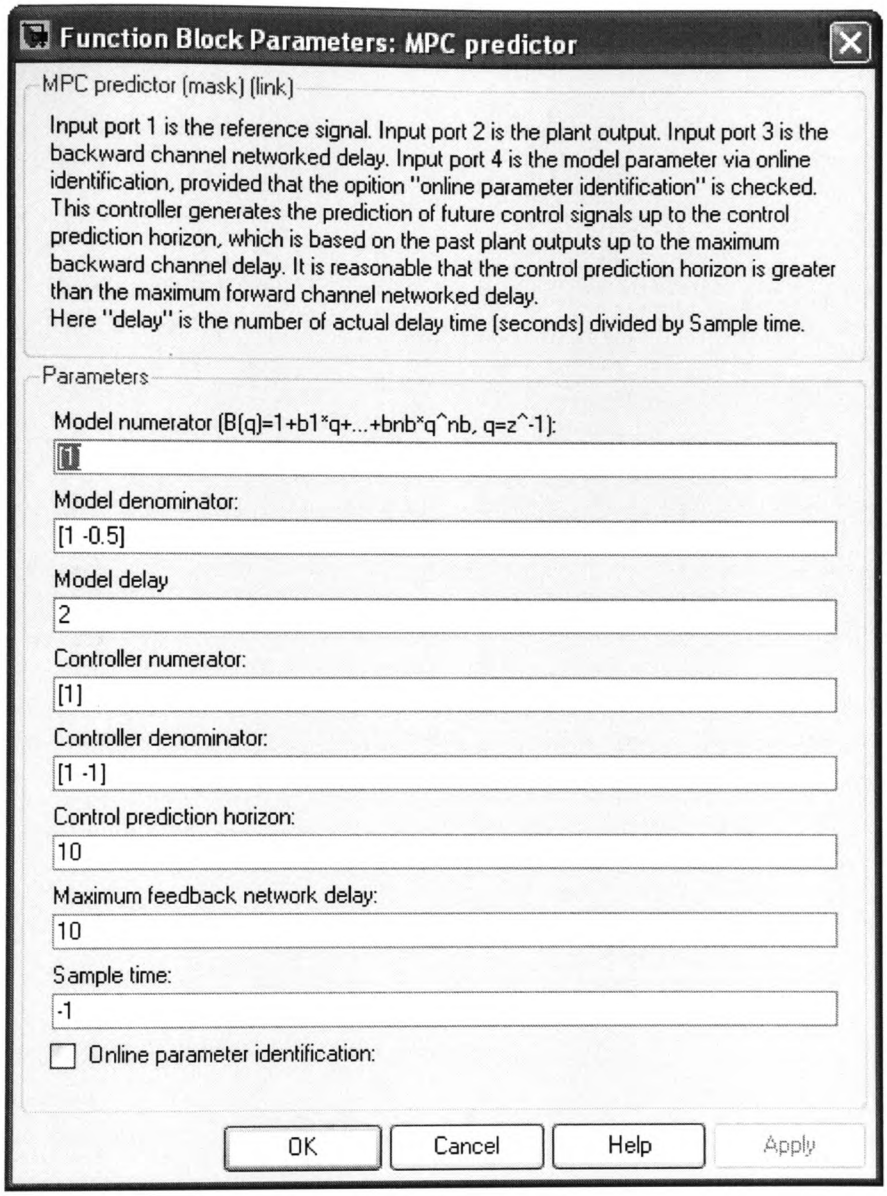



Figure A.4: Configuration of control scheme of MPC predictor.